2018

Machine Learning for Exploring State Space Structure in Genetic Regulatory Networks

Rodney H. Thomas
Nova Southeastern University, rt362@mynsu.nova.edu

Machine Learning for Exploring State Space Structure in Genetic Regulatory Networks

by

Rodney Hartfield Thomas

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
Computer Science

College of Engineering and Computing
Nova Southeastern University

June 2018

We hereby certify that this dissertation, submitted by Rodney Thomas, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

Michael J. Laszlo, Ph.D.
Chairperson of Dissertation Committee

6/6/18
Date

Francisco J. Mitropoulos, Ph.D.
Dissertation Committee Member

6/8/2018
Date

Sumitra Mukherjee, Ph.D.
Dissertation Committee Member

6/8/2018
Date

Approved:

Yong X. Tao, Ph.D., P.E., FASME
Dean, College of Engineering and Computing

6/8/2018
Date

College of Engineering and Computing
Nova Southeastern University

2018

An Abstract of a Dissertation Proposal Submitted to Nova Southeastern University
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Machine Learning for Exploring State Space Structure in Genetic Regulatory Networks

by
Rodney Hartfield Thomas
June 2018

Genetic regulatory networks (GRN) offer a useful model for clinical biology. Specifically, such networks capture interactions among genes, proteins, and other metabolic factors. Unfortunately, it is difficult to understand and predict the behavior of networks that are of realistic size and complexity. In this dissertation, *behavior* refers to the trajectory of a state, through a series of state transitions over time, to an attractor in the network. This project assumes *asynchronous* Boolean networks, implying that a state may transition to more than one attractor. The goal of this project is to efficiently identify a network's set of attractors and to predict the likelihood with which an arbitrary state leads to each of the network's attractors. These probabilities will be represented using a fuzzy membership vector.

Predicting fuzzy membership vectors using machine learning techniques may address the intractability posed by networks of realistic size and complexity. Modeling and simulation can be used to provide the necessary training sets for machine learning methods to predict fuzzy membership vectors. The experiments comprise several GRNs, each represented by a set of output classes. These classes consist of thresholds τ and $\neg\tau$, where $\tau = [\tau_{low,}\tau_{high}]$; state s belongs to class τ if the probability of its transitioning to attractor A belongs to the range $[\tau_{low,}\tau_{high}]$; otherwise it belongs to class $\neg\tau$. Finally, each machine learning classifier was trained with the training sets that was previously collected. The objective is to explore methods to discover patterns for meaningful classification of states in realistically complex regulatory networks.

The research design took a GRN and a machine learning method as input and produced output class $<A, \tau>$ and its negation $\neg<A, \tau>$. For each GRN, attractors were identified, data was collected by sampling each state to create fuzzy membership vectors, and machine learning methods were trained to predict whether a state is in a healthy attractor or not. For T-LGL, SVMs had the highest accuracy in predictions (between 93.6% and 96.9%) and precision (between 94.59% and 97.87%). However, naive Bayesian classifiers had the highest recall (between 94.71% and 97.78%). This study showed that all experiments have extreme significance with $p_{value} < 0.0001$. The contribution this research offers helps clinical biologist to submit genetic states to get an initial result on their outcomes. For future work, this implementation could use other machine learning classifiers such as xgboost or deep learning methods. Other suggestions offered are developing methods that improves the performance of state transition that allow for larger training sets to be sampled.

Keywords: asynchronous Boolean networks, attractors, Boolean networks, cross-validation, decision trees, fuzzy basins, fuzzy membership vectors, fuzzy vectors, genetic regulatory networks, Markov Chain Monte Carlo, naïve Bayesian classifiers, support vector machines

Acknowledgements

I would first like to thank my dissertation advisor Professor Michael Laszlo of the College of Engineering and Computing at Nova Southeastern University. The door to Professor Laszlo's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to thank the committee members who were involved with this research project: Professor Sumitra Mukherjee and Professor Francisco Mitropoulos. Without their passionate participation and input, this dissertation report could not have been successfully completed.

I would also like to acknowledge Doctor Caroline Eisner of Middlebury College as the second reader of this dissertation report, and I am gratefully indebted to her for her very valuable comments on this report.

Nobody has been more important to me in the pursuit of this project than the members of my family. I wish to thank my loving and supportive wife, Colleen, and my daughter, Gemma, who provide unending inspiration.

Table of Contents

List of Tables

Tables

List of Figures

Figures

Chapter 1

Introduction

Background

Different types of diseases are caused by a blend of genetic inheritance, environmental influence, and lifestyle choices. These diseases have features or symptoms that can indicate whether they exist within a subject. Implementing a genetic screening process is a possibility for using these symptoms to discover the cause of genetic and genomic patterns. Understanding the cause of these patterns can increase the chances of finding cures and answering potential research questions. Research interests include single genetic disorders to polygenic connections.

This research will focus on algorithms that will simulate cancerous white blood cells, mammalian cell cycle, cardiac development, immune Bb, and Abscisic acid signaling model. Numerous algorithms can be used to solve these problems using graph-based techniques. Some classes of techniques that are currently being used are coupled ordinary differential equations, Boolean networks, continuous networks, and stochastic gene networks. Boolean networks will be explored as the graph-based technique for this study. This technique can simulate a higher level concept known as *genetic regulatory networks* (GRN). It is important to use a GRN and its functional biology for the identification of mechanisms of complex diseases and therapeutic targets.

A GRN represents the interactions among DNA segments of a cell that interact with each other indirectly and with other cellular particles and thereby govern the expression levels of mRNA and proteins. Genetic regulation guides the division and differentiation of cells from progenitor cells (stem cells) into mature cells, temporal and spatial movement, and the

termination of a cell within the organism (Hallinan, 2006). GRN seeks to describe genes or groups of genes interaction with each other and identify the complex regulatory mechanisms that control the activities of genes in living cells (Jong, 2002). These networks can help life scientists understand the intricate interactions of multiple genes under different environmental conditions (Cao, 2010). GRNs also help life scientists understand interactions between the molecules in a cell's functions and reactions to levels of treatment, and the logical structures of functionality within a cell.

An example of a GRN is a genetic *cell cycle regulatory network* (cell cycle) of animals. Cell cycle is the process of a cell doubling its genetic content using a set period and distributing those contents to two daughter cells after splitting. In most cells, the time is coupled with the duplication of other cell contents, and cells can divide only after doubling their size (Sveiczer, 1996). Unwanted cells, such as cancer, also follow the cell cycle process. Cancer cells grow at abnormal rates and cause harm or termination to an organism. Applying a specific treatment for a cancerous cell cycle could force the cell to terminate. The application of new treatments using trial and error to an organism also could cause it harm or death. Thus, modeling and simulating GRNs with accurate predictions could eliminate random trials of treatments. Therefore, the implementation and execution of *random Boolean networks* as a model for cell cycle and other GRNs could simulate life systems and the effects of treatments once it is applied to them.

A *Boolean network* can be used to model and simulate a GRN, including its gene transcription factors that serve as outputs, and gene expression functions that serve as inputs. Living organisms could be constructed from a random Boolean networks without the need of precisely programmed elements (Kauffman, 1969). A GRN has a gene, protein, or chemical compound that is represented by a node in a directed graph in which there is activation or

inhibition from one node to another if and only if there is a causal link between the two nodes. A Boolean network is a set of nodes, corresponding to variables. Each node's value depends on a Boolean function of the activator nodes and inhibitor nodes that are this node's input.

This study will use the terms *variable* and *node* interchangeably with Boolean networks. Each node in the network can be in the *on (true)* or *off (false)* state. The *on* state corresponds to the gene being expressed and the *off* state corresponds to it not being expressed. The state of the network comprises the state of its nodes at a given time instance. Time will increment in discrete steps updating variables in the process. At each discrete step, a Boolean function will cause a transition from one state to a new state. By comparing simulation results from time series observations, the Boolean network model can be validated.

Gene expression is a tightly regulated process that allows a cell to respond to its changing environment. This is the process by which instructions in our DNA are converted to a protein or other biological molecules (Hallinan, 2006). It acts as both an on or off switch to control when proteins are made and also a volume control that increases or decreases the amount of proteins made. Boolean networks can simulate this biological process by acting as the on or off which represents whether a gene in a GRN is being expressed or not expressed respectively.

A Boolean network can be considered as a directed graph $G = < V, E >$. The network consists of a set of variables $X = \{x_1, \dots, x_n\}$, and a set of corresponding transition functions $F = \{f_1, \dots, f_n\}$. Let D_i denote the co-domain of variable i, and function $f_i: D_1 \times \dots \times D_n \to D_i$, where D_i is $\{0,1\}$ for Boolean networks. Each node $v_i \in V$ has an associated state variable $x_i \in \{0,1\}$ and a state transition function $f_i: \{0,1\}^n \to \{0,1\}$. The presence of edge $e_{ij} = < v_i, v_j > (i, j \in \{1, \dots, n\})$ directed from node v_i to node v_j indicates that the next state of node v_j depends on the current state of node v_i. The behavior of a network over time is governed by its

transition functions. The state of the network at time t is a vector $\overrightarrow{x_t} = \left(x_1(t), \dots, x_n(t)\right)$, where $x_i(t) \in D_i$. In a *synchronous* network, the successor state $\overrightarrow{x_{t+1}}$ of $\overrightarrow{x_t}$ is given by $\left(f_1(x_t), \dots, f_n(x_t)\right)$. In an *asynchronous* network, the successor state $\overrightarrow{x_{t+1}}$ of $\overrightarrow{x_t}$ differs from $\overrightarrow{x_t}$ in the value of at most one variable X_{i^*}. Specifically, $x_{i^*}(t+1) = f_{i^*}(\overrightarrow{x_t})$, and $x_i(t+1) = x_i(t)\ \forall\ i \neq i^*$, where i^* is selected based on a specified probability distribution. In this research, the *HPA-GR-Immune-HPGa for males* GRN in Figure 1 is modeled and simulated by the *HPA-GR-Immune-HPGa for males* asynchronous Boolean network in Figure 2. These two figures show the same network diagrammatically (Figure 1) and textually using the input file format (Figure 2).

Asynchronous Boolean networks have an advantage over synchronous Boolean networks insofar as they model real regulatory systems more realistically. In a GRN simulated by a synchronous network, there are no overlapping attractors, in which a given start state s will always transition to exactly one attractor. In a clinical setting, no treatment is guaranteed to work 100% of the time for all known patients in need of a cure, due to random as well as poorly understood factors involved. However, using an asynchronous Boolean network can simulate this reality. Furthermore, asynchronous networks do have overlapping attractors and a start state s could sometimes transition into one attractor or the other.

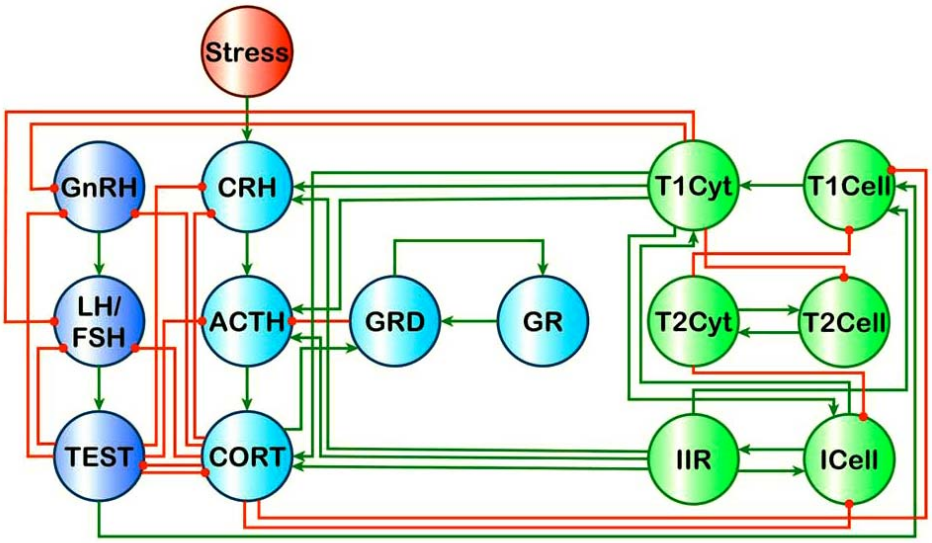

Figure 1: HPA-GR-Immune-HPGa GRN. A green edge and arrowhead indicate activation and a red edge and point indicate inhibition (Craddock et al., 2014).

HPA-GR-Immune-HPGa Boolean Network

```
Stress  *= Stress
CRH     *= Stress ∧ ¬CORT ∧ Th1Cyt ∧ IIR ∧ ¬TEST
ACTH    *= CRH ∧ ¬GRD ∧ Th1Cyt ∧ IIR ∧ ¬TEST
CORT    *= ACTH ∧ Th1Cyt ∧ IIR ∧ ¬TEST
GRD     *= CORT ∧ GR
GR      *= GRD
ICell   *= ¬CORT ∧ IIR ∧ Th1Cyt ∧ ¬Th2Cyt
IIR     *= ICell
Th1Cell *= IIR ∧ ¬Th2Cyt ∧ Th1Cyt ∧ TEST
Th1Cyt  *= ICell ∧ Th1Cell
Th2Cell *= ¬Th1Cyt ∧ Th2Cyt
Th2Cyt  *= Th2Cell
GnRH    *= ¬TEST ∧ ¬CORT ∧ ¬Th1Cyt
LH/FSH  *= ¬TEST ∧ GnRH ∧ ¬CORT ∧ ¬Th1Cyt
TEST    *= LH/FSH ∧ ¬CORT
```

Figure 2: HPA-GR-Immune-HPGa Boolean Network (Craddock et al., 2014).

In the format of Figure 2, each line has a left-hand side, a right-hand side that represents the transition function, and the equality symbol *= that separates both sides. The left-hand side is a set of Boolean variables that will be updated asynchronously by the Boolean expression on the right-hand side. The Boolean variables correspond to genes, proteins, and so forth, with the GRN from Figure 1. The Boolean operators appearing in the right-hand sides are the *and* symbol, represented by ∧; the *or* symbol, represented by ∨; and the *not* symbol, represented by ¬. Figure 2 does not use the *or* symbol, but it is present in many Boolean networks.

Chosen at random, a state transition is performed by choosing one of the node's coordinate functions and updating its corresponding Boolean variable to the current state for asynchronous Boolean networks. This corresponds to the assumption that, in a genetic network, gene expression levels are likely to change at different points of time (Thomas, 1991). The importance of this modeling technique is that it resembles a Markov Chain, and during sampling, the Markov Chain Monte Carlo method can be used to collect data to determine fuzzy membership vectors. There is a noticeable transition difference between synchronous and asynchronous networks. Every state in a synchronous network can only transition into one other state. However, every state in an asynchronous network can transition into multiple states which can lead to overlapping attractors.

Boolean networks have 2^n states, where n is the total number of nodes in the network, ensuring that a transition sequence must eventually revisit some states. These transitions will fall into a cycle of visiting previously visited states. Such cycles are known as attractors (Hopfensitz, Müssel, Maucher, & Kestler, 2013). Markov Chains have a similar phenomenon. A Markov Chain that has cycles is known as an Absorbing Markov Chain. For instance, for a state s, the set of *reachable* states $R(s)$ is defined to be the set of all states that can be reached from s through a finite sequence of state transitions. A set of states $S = \{s_1, \dots, s_k\}$ is said to be an *attractor* if for every $s_i \in S$, $R(s_i) = S$. When $R(s) = \{s\}$, s is called a *steady state*. All states that lead to an attractor are part of that attractor's basin. In asynchronous Boolean networks, some states may lead to more than one attractor; those states will have fuzzy membership in each attractor due to the overlapping nature of attractors in asynchronous networks.

Let Y represent a sequence of transitions starting from state s_i leading to attractor A, with a list of elements for Y denoted by y_i. Thus $Y = y_1, \dots, y_n$. A fuzzy set $\tilde{A}$ in Y is characterized by

a membership function $\mu_{\tilde{A}}(y_i)$ which associates with each element in Y a real number in the interval $[0, 1]$, with the values of $\mu_{\tilde{A}}(y_i)$ at y_i representing the *degree of membership* of y_i in $\tilde{A}$ (Zadeh, 1965). Thus, the nearer the value of $\mu_{\tilde{A}}(y)$ to attractor A, the higher the degree of membership of y_i in $\tilde{A}$. Thus, the value 0 means that y_i is not a member of the fuzzy set and the value 1 means that y_i is fully a member of the fuzzy set. Values that lie strictly between 0 and 1 characterize fuzzy membership, which belong to the fuzzy set only partially and the set of states S resides in the fuzzy basin where two or more attractor basins overlap. The final result is an approximation of the exact fuzzy membership vector.

The exact fuzzy membership vector in its canonical form is the transition probability matrix, $P = \begin{bmatrix} Q_t & R \\ 0 & I_r \end{bmatrix}$. Q_t is a $t\ x\ t$ matrix of transient states and t is the number of existing transient states. A transient state is a state that is not part of the list of attractor states. I_r is an $r \times r$ identity matrix and r is the number of existing attractor states. R is a non-zero $t \times r$ matrix which represents the probability of a transient state transitioning to an attractor state. Finally, 0 is an $r \times t$ zero matrix.

To determine the probability of reaching attractor state s_j when starting from a transient state s_i, the probability matrix $B = NR$ has to be calculated. R is a non-zero $t \times r$ matrix stated above and $N = \sum_{k=0}^{\infty} Q^k = (I_t - Q)^{-1}$. A problem with using this canonical form is calculating all of the probabilities for a very large state space such as *T-cell large granular lymphocytic leukemia* (T-LGL). Thus, approximating the fuzzy membership vectors can be done in acceptable time or in deterministic polynomial time. Therefore, this research will continue using asynchronous Boolean networks.

To identify that a state has led to an attractor, an algorithm for searching attractors beforehand must be performed. A straightforward algorithm to identify attractors in

asynchronous Boolean networks starts from a set of start states and repeatedly performs state transitions until a forward-set of states has been validated as an attractor (Hopfensitz et al., 2013). There can be more than one successor state for each individual state within an asynchronous Boolean network. The asynchronous attractor search heuristic begins at a start state and applies transitions until an attractor is reached. The distribution used to select the successor state in an asynchronous network is determined by $x_{i^*}(t+1) = f_{i^*}(x_t)$ where i^* is a randomly selected node of the current state.

Asynchronous attractor search algorithm

Input: A Boolean network with n genes and n asynchronous state transition functions
$\boldsymbol{T}_{async}^{(i)}: B^n \rightarrow B^n$
A number of random transitions r
A set of m start states, $S = \{(s_{11}, \dots, s_{1n}), \dots, (s_{m1}, \dots, s_{mn})\}$
$resultList \leftarrow \emptyset$
for all $startState \in S$ **do** $currentState \leftarrow startState$
for $i = 1, \dots, r$
do {Perform a random asynchronous state transition on $currentState$}
$i^* \leftarrow rand(n)$
$currentState \leftarrow \boldsymbol{T}_{async}^{(i^*)}(currentState)$
end for
$attractor \leftarrow ForwardSet(currentState)$
if $ValidateAttractor(attractor)$ **then** {This is a true attractor}
$resultList \leftarrow resultList \cup \{attractor\}$
end if
end for
return $resultList$

Figure 3: Asynchronous attractor search algorithm (Hopfensitz et al., 2013).

Function ForwardSet
Input: A state *s* for which the forward reachable set is determined
$resultSet \leftarrow \{s\}$
$stack \leftarrow \{s\}$
repeat
  $current \leftarrow pop(stack)$
  for $i = 1, \ldots, n$
  do {Calculate successor states}
    $next \leftarrow \boldsymbol{T}^{(i)}_{async}(current)$
    if $(next \notin resultSet)$ **then**
      $resultSet \leftarrow resultSet \cup \{next\}$
      $push(stack, next)$
    end if
  end for
until $(stack = \emptyset)$

Figure 4: Forward Set Function (Hopfensitz et al., 2013).

Function ValidateAttractor
Input: A set of states *S* to be validated
for all $s \in S$ **do**
  if $(ForwardSet(s) \neq S)$ **then**
    return false
  end if
end for
return true

Figure 5: Validate Attractor Function (Hopfensitz et al., 2013).

States that reside in the fuzzy basin are sets whose elements have fuzzy degrees of membership in multiple attractors. Fuzzy classification is the process of grouping elements into a fuzzy set (Zadeh, 1965) whose degree of membership is defined by the ratio of the number of times a state transitions into an attractor to the total times a simulation was executed. Understanding fuzzy degree of membership for individual states allows scientists to use exploratory data analysis techniques to see if there are any structures within the fuzzy basin area such as clustering or other patterns that would be useful to life scientists.

The graph in Figure 6 shows data collected using the asynchronous Boolean network from Figure 2. The *stress* variable is set to *true* and remains a constant throughout the simulation.

In addition, all of the states were part of the sampling process because this is a small network. There are 2^{14} states or 16384 states, and each state was sampled N=1000 times. The state will transition into either attractor A or attractor B. The graph shows the bin count that a state will transitions into attractor B out of 1000 trials. Not shown in Figure 6 are the states in the true basins of attraction which appear in B either zero or 1000 times. There are 1536 states that appear in B zero times and 4096 states that appear in B 1000 times. However, this data was purposely left out of the graph. In the graph, the bars for these two instances would have been much taller than the bar which represents 89 states that appear in B 500 times, currently the tallest bar in Figure 6. Leaving these two instances in the graph would give a flatten appearance and it would be difficult to visualize the fuzzy structure. This preliminary experiment exhibits *fuzzy membership structure* in this small asynchronous Boolean network.

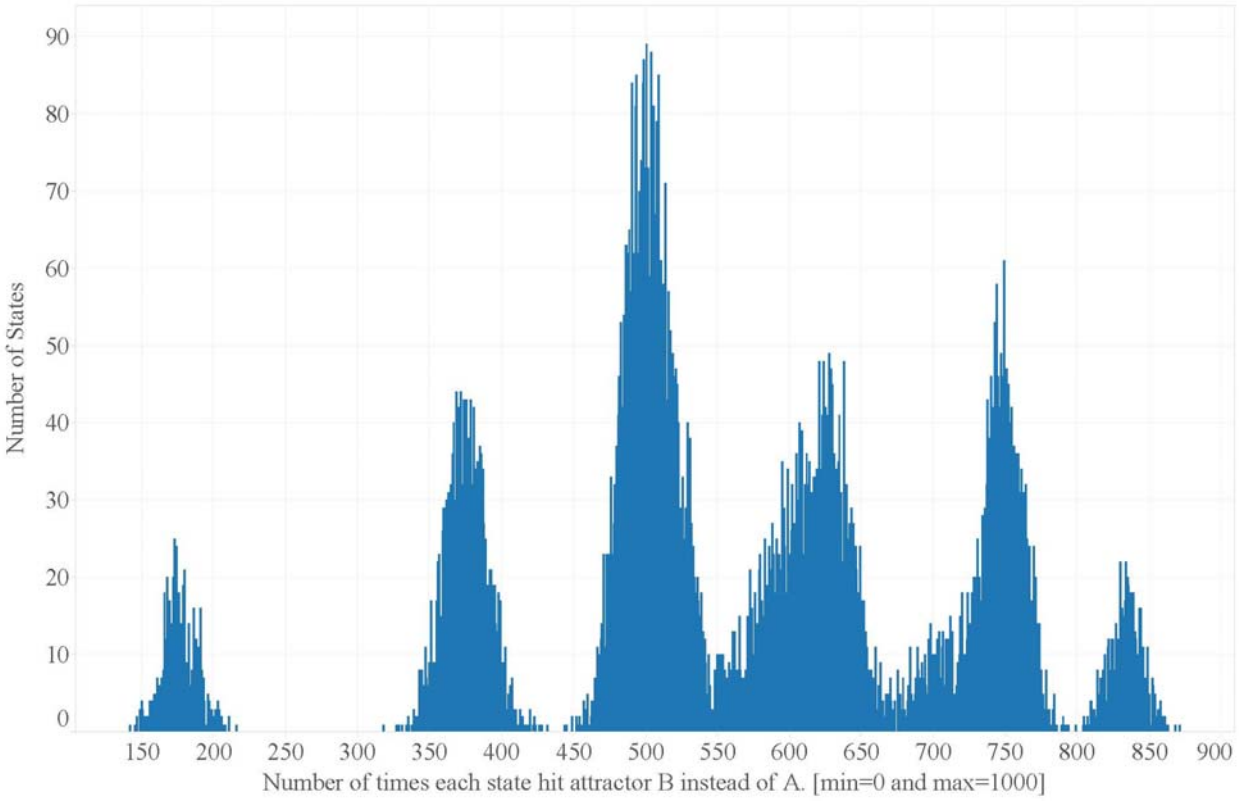

Figure 6: Fuzzy Membership for Attractor B. This Figure does not show the states that hit attractor B zero times or the hits to attractor B every time.

Markov chain analysis has methods for creating a transition matrix that, for this study, would be a matrix with the probability of a state transitioning into a set of attractors. A transition matrix is used to describe the transitions of a Markov chain (Asmussen, 2003). Each of its entries is a nonnegative real number representing a probability. Unfortunately, for large and even moderate size networks, transition matrices take up large amounts of resources and are very expensive to compute. For this study, creating transition matrices for Boolean networks is intractable. Therefore, this report will not use transition matrices. However, Markov Chain Monte Carlo methods will be used for collecting data.

Problem Statement and Dissertation Goals

Due to the size and complexity of realistic GRNs, it is infeasible to use simulation to estimate fuzzy membership vectors for all states. The goal of this dissertation is to use simulation

to determine fuzzy membership vectors for a select sample of states, and then to use machine learning to discover patterns in membership that might extend to a much larger range of, or perhaps all, states. These patterns, exemplified by rules, would yield fuzzy membership vectors for states not subject to simulation. The machine learning methods to be employed are decision trees, SVMs, and naïve Bayesian classifiers. The training data for these classifiers will be obtained using Markov Chain Monte Carlo sampling, yielding fuzzy membership vectors. The ideas presented here are predicated on the existence of structure in overlapping attractors; early experiments have indicated that such structure often exists.

Research Questions

Preliminary work shows that HPA-GR-Immune-HPGa Boolean Network has fuzzy membership structure for its state space. Do realistic GRNs such as T-LGL possess membership structure? What is the nature of this structure?

Realistic GRNs have many nodes and finding the membership for all or many states is intractable. Can machine learning methods be used to discover such structure? In practical terms, can machine learning methods be used to identify accurate fuzzy membership vectors for states?

Three machine learning techniques will be used to make predictions of random states. The machine learning algorithms to be in usage are support vector machines, naive Bayesian classifier, and decision trees. Which machine learning methods, of those considered, are most effective?

What are the most useful output classes for this application of machine learning methods? Examples are (a) finding a state belonging (or not) to an attractor's true basin; and (b) for a given attractor A and a specified threshold range τ, the output class of a state is positive if the probability of reaching attractor A starting from the state lying within the specified range τ.

Relevance and Significance

With very large GRNs, it is not feasible to determine the fuzzy membership for all states using simulation. A large GRN is modeled and simulated by an asynchronous Boolean network that could have overlapping basins of attraction. Once the system can accurately predict the basins of attraction to which a state belongs, life science experts can then predict the results of different treatment regimens and see immediate results. This will allow experts to make critical decisions and avoid applying a physical treatment directly to their subjects using trial and error.

Barriers and Issues

In very large realistic GRNs such as T-LGL, finding the basin of attraction and fuzzy membership in that basin is an intractable task. Having 32 nodes pushes the upper limit of what a physical machine can do to find attractors and their basins. However, T-LGL has 60 nodes, and it would require a large cluster of machines at high cost to find all of the states' fuzzy membership in attractors (Hong et al, 2015). Therefore, using machine learning to predict only the states that are important to an expert's experiment is a feasible option.

The algorithms in Figures 3, 4, and 5 are presented partly for defining operationally the meaning of identifying and validating attractors. These algorithms may be used, as they are in this research, but where necessary, more efficient stochastic approaches could be used. There is a significant amount of literature based on efficiently finding attractors; thus, exploring previous work will help with this issue.

Assumptions, Limitations, and Delimitations

This research is based on several assumptions. First, there is the assumption that a structure exists in the state space of each GRN. This structure is necessary to determine the output classes. Second, states with the same output classes can transition into each other. For instance, suppose that the state-space of a Markov chain is divided into a disjointed subset of

states. Let t_i be a subset of $S = \{s_1, \dots, s_k\}$ and $T = \{t_1, \dots, t_n\}$ is the partition of all states. Thus, all states within subset t_i have a higher probability of transitioning into each other than with states in another subset. These clusters of states will lead to the same output classes. Finally, during the simulation process, attractors can be discovered quickly.

This research also has some limitations. One limitation is that it is not known if one cluster of states is actually a multiple number of sub-clusters and these sub-clusters are possibly not interconnected. Several factors are under control in this research. Five GRNs were selectively chosen. Also, three classifiers were selectively chosen. Furthermore, studying the performance of a statistical approach to determine the outcome of a state is a problem chosen for this research.

Summary

Genes regulate each other's activity through GRNs. Part of the gene expression process is transcription in which a particular segment of DNA is copied into mRNA. Furthermore, post-translational modifications lead to proteins with modified properties. When attempting to model and simulate a GRN, the first step is to assemble the components of the network and the interactions between them. For this research, *asynchronous Boolean networks* will serve as that paradigm.

A complete GRN model incorporates experimental knowledge about the components and their interactions as well as the initial state of these components. This leads to the known final state of attraction or dynamical behavior of the network. Validated models such as cell cycle or T-LGL are able to investigate cases that cannot be explored experimentally, for example changes in the initial state, in the components, or in the interactions, and these models can lead to predictions and insights into the functioning of the system.

The research questions ask about state space structures and their effects on a state's fuzzy membership vector to determine output classes. Also, the questions ask about which machine learning techniques are best to determine the output class of a state. Finally, the question asks what the best output classes are. This can be determined by running simulations on a subset of states, creating a visualization of fuzzy membership to an attractor to identify structure, implementing a threshold range on possibly the best cluster, train a machine learning classifier to predict whether a state is in that cluster or not in that cluster, and measure that accuracy of the prediction.

Chapter 2

Review of the Literature

Overview of the Literature

The synchronous Boolean update model has the expression level of *every* gene updated at discrete time points. However, some genes retain their previous level under the transition function. This approach is computationally tractable for very large networks but it does not accurately represent different genes transitioning from one expression level to another at discrete time intervals. Thus, synchronous Boolean networks are biologically imperfectly realistic. Prior work on Boolean networks has focused primarily on synchronous update procedures which include the analysis of the attractor's state space and the basin of attraction for each attractor (Faure, Naldi, Chaouiya, & Thieffry, 2006). The advantage of using synchronous update models is in finding steady state attractors quickly in an exhaustive search. The steady state attractors found in synchronous searches also exist in the asynchronous update model.

A large portion of current studies involves asynchronous Boolean models and have focused mainly on finding attractors of a system or on identifying attractors reachable from a nominal initial state or condition (Chaves, Albert, & Sontag, 2005). The asynchronous Boolean update model has exactly one gene's expression level changing at each discrete time point. The gene is selected at random from among those whose expression levels would change under the transition function. This update model closely represents biological activity but is more complex to model and analyze. A state could potentially reside in the basin of attraction of multiple attractors. For example, a state s could transition into attractor A in one sampling and transition into attractor B in another sampling. For this state s, the two attractors overlap.

Few studies focus on identifying complex and overlapping attractors (Garg, Di Cara, Xenarios, Mendoza, & De Micheli, 2008). Thus, the investigation of all possible attractors, their basins of attraction, and fuzzy membership of states that fall into overlapping attractors for a system under asynchronous updating schemes invites further research. This research will perform a comprehensive study of all attractors of a large biological system, focusing on *T-cell large granular lymphocytic leukemia* (T-LGL), selecting a number of initial states at random, and using methods or performing simulations to determine the fuzzy membership of those states using the asynchronous Boolean network model. This will identify the structures of the network or the fuzzy membership clusters, which will assist in making predictions for states not in the initial sampling process.

T-LGL leukemia affects lymphocytes (white blood cells) which are part of the body's immune system and help fight infections. Some signs and symptoms of T-LGL are changes in blood cell counts, decline in production of red blood cells, recurrent infections, and fever. Diagnose can be confirmed by examining the patient's blood under a microscope by checking to see if a large number of abnormal cells associated with T-LGL may be present, or by taking a bone marrow aspiration, or biopsy. Some treatments currently in practice are immunosuppressive therapy, such as methotrexate, oral cyclophosphamide (an alkylation agent), and cyclosporine (an immunomodulatory drug).

The cell cycle process will respond to external environmental changes in order to maintain their functional purpose such as growth, survival, division, and apoptosis (cell death). This *gene regulatory network* (GRN) process is carried out through a chain reaction of gene, protein, and chemical interactions forming a complex signaling network. An abnormal expression between some of the components in the network will affect normal operations of the

cell, which will transition it from a healthy state to a poor state, translating to possible diseases such as diabetes (Leibiger, Brismar, & Berggren, 2010), developmental disorders (Gordon & Blobe, 2008), autoimmunity (Mavers, Ruderman, & Perlman, 2009), or cancer (Ikushima & Miyazono, 2010).

Healthy *cytotoxic T-lymphocytes* (CTLs) are produced to eliminate cell infections of viruses. However, healthy CTLs will initiate apoptosis after they successfully destroy the infected cell; but, leukemic T-LGL cells fail to initiate apoptosis and remain in the system for long periods (Sokol & Jr, 2006). A few components of the CTLs are responsible for the abnormal behavior of the signal transduction network which activates apoptosis for T-cells (Shah, Zhang, & Jr, 2009). Therefore, understanding which states lead to a healthy attractor, such as apoptosis in T-LGL, will suggest treatments that switch from one state to a nearby state that tends to lead to a healthy attractor.

A Boolean network model of T-cell survival signaling in the context of T-LGL leukemia was implemented by Zhang et al. (Zhang, Shah, Yang, SB, & X, 2008). The implementation was guided by performing an extensive literature search on the topic, compiling the data, and finding the Boolean equivalent functions to match the results of the data. The T-LGL network consists of 60 components including receptors, proteins, mRNAs, and small molecules. The network contains six nodes with no upstream components representing external input signals (Stimuli, IL15, PDGF, Stimuli2, CD45, and TAX), and also contains three output nodes serving as indicators of biological functions or cell fate (Cytoskeleton signaling, Proliferation, and Apoptosis). The main input to the network is *Stimuli*, which represents a virus or antigen stimulation, and the main output node is *Apoptosis*, which represents programmed cell death (Zhang et al., 2008).

After implementing the T-LGL network, the methods of asynchronous Boolean network exposed a small number of impairments that led to the cause of T-LGL survival, showing high activity within proteins *platelet-derived growth factor* (PDGF) and interleukin 15 (IL15) (Zhang et al., 2008). The preliminary analysis of T-LGL network dynamics was carried out by performing numerical simulations starting from one specific condition in which the T-Cell receives a stimuli and an overload of two proteins PDGF and IL15 (Zhang et al., 2008). Once the issues of T-LGL leukemia was identified, each of these symptoms was interrupted individually by reversing the node's state in order to predict key mediators of the disease (Zhang et al., 2008). However, a complete dynamic analysis of the system, its corresponding basins of attraction, as well as a thorough perturbation analysis of the system considering all possible initial states was not undertaken. Though, the attractors of T-LGL has been identified in previous experiments, and the results are validated by other researchers (Zanudo & Albert, 2013). Implementing machine learning methods to accurately predict fuzzy membership within the basins of attractions for initial states can provide deeper insights into unknown aspects of T-LGL leukemia. For this research, the predictive analysis of states relies on consistent detection of clusters. These clusters will be used to train classifiers to determine if a state is in or not in a specific cluster. This process will determine the fuzzy membership vectors for states. The modeling and simulation of a GRN will be used to collect data and classifiers can be trained by that data which will produce an optimize process to predict the outcome of states that was not part of the initial selection.

Tables 1 and 2 show the attractors found in T-LGL leukemia GRN. The findings from the experiments used to obtain these results are an exact match with the findings from Zanudo & Albert, 2013. Columns A, B, C, and D represent the attractor findings based on different external

inputs values from Stimuli, IL15, PDGF, Stimuli2, CD45, and TAX. Each column overlaps with the Apoptosis attractor for some individual start states. An external input value set to *on*/*off* means that the external input node does not affect the outcome of finding that particular attractor. Also, in columns A and C, when a state *s* transitions into the attractor, P2 could be *on* in one sampling and *off* on a different sampling which indicates two different attractors. Therefore, for those columns, P2 is set to ON|OFF to indicate this particular sampling effect. Finally, all of the nodes in a column marked with oscillation are nodes that make up the attractor.

Attractors	A	B	C	D	Apoptosis
INPUT NODES					
CD45	ON	ON	OFF	OFF	ON/OFF
PDGF	ON/OFF	ON/OFF	ON/OFF	ON/OFF	ON/OFF
IL15	ON/OFF	ON/OFF	ON	ON	ON/OFF
Stimuli	ON	ON	ON	ON	ON
Stimuli2	OFF	ON	OFF	ON	ON/OFF
TAX	ON/OFF	ON/OFF	ON/OFF	ON/OFF	ON/OFF
NODES					
IL2RBT	OFF	OFF	ON	ON	OFF
BclxL	ON	ON	OFF	OFF	OFF
IFNGT	ON	ON	ON	ON	OFF
PDGFR	ON	ON	ON	ON	OFF
IFNG	OFF	OFF	OFF	OFF	OFF
GAP	OFF	OFF	OFF	OFF	OFF
Proliferation	OFF	OFF	OFF	OFF	OFF
GZMB	OFF	OFF	ON	ON	OFF
RAS	ON	ON	ON	ON	OFF
TPL2	ON	ON	ON	ON	OFF
FasT	ON	ON	ON	ON	OFF
FLIP	ON	ON	ON	ON	OFF
LCK	ON	ON	ON	ON	OFF
NFAT	ON	ON	ON	ON	OFF
FasL	ON	ON	ON	ON	OFF
Caspase	OFF	OFF	OFF	OFF	OFF
NFKB	ON	ON	ON	ON	OFF
IAP	ON	ON	ON	ON	OFF
BID	OFF	OFF	OFF	OFF	OFF
Cyto. Signal.	Oscillates	Oscillates	ON	ON	OFF
TNF	ON	ON	ON	ON	OFF
MCL1	OFF	OFF	ON	ON	OFF
Ceramide	OFF	OFF	OFF	OFF	OFF
GRB2	Oscillates	Oscillates	ON	ON	OFF
PI3K	ON	ON	ON	ON	OFF
SMAD	ON	ON	ON	ON	OFF
P27	OFF	OFF	ON	ON	OFF
ZAP70	Oscillates	Oscillates	OFF	OFF	OFF
CREB	OFF	OFF	OFF	OFF	OFF
DISC	OFF	OFF	OFF	OFF	OFF
IL2RB	OFF	OFF	ON	ON	OFF
Fas	OFF	OFF	OFF	OFF	OFF
IL2RA	Oscillates	Oscillates	OFF	OFF	OFF
S1P	ON	ON	ON	ON	OFF
ERK	ON	ON	ON	ON	OFF
SPHK1	ON	ON	ON	ON	OFF
A20	ON	ON	ON	ON	OFF
MEK	ON	ON	ON	ON	OFF
CTLA4	Oscillates	Oscillates	Oscillates	Oscillates	OFF
TBET	OFF	OFF	ON	ON	OFF
RANTES	ON	ON	ON	ON	OFF
SOCS	OFF	OFF	OFF	OFF	OFF
sFas	ON	ON	ON	ON	OFF
IL2RAT	ON	ON	OFF	OFF	OFF
TCR	Oscillates	Oscillates	Oscillates	Oscillates	OFF
STAT3	OFF	OFF	ON	ON	OFF
GPCR	ON	ON	ON	ON	OFF
P2	ON|OFF	OFF	ON|OFF	OFF	OFF
TRADD	OFF	OFF	OFF	OFF	OFF
PLCG1	ON	ON	ON	ON	OFF
FYN	Oscillates	Oscillates	ON	ON	OFF
IL2	ON	ON	OFF	OFF	OFF
JAK	OFF	OFF	ON	ON	OFF
Apoptosis	OFF	OFF	OFF	OFF	ON

Table 1: The attractors of T-LGL leukemia survival network. This table shows the state of the nodes for all possible combinations of input signals in the presence of antigen (Stimuli=ON).

Attractors	A	B	C	D	Apoptosis
INPUT NODES					
CD45	ON	ON	OFF	OFF	ON/OFF
PDGF	ON/OFF	ON/OFF	ON/OFF	ON/OFF	ON/OFF
IL15	ON/OFF	ON/OFF	ON	ON	ON/OFF
Stimuli	OFF	OFF	OFF	OFF	OFF
Stimuli2	OFF	ON	OFF	ON	ON/OFF
TAX	ON/OFF	ON/OFF	ON/OFF	ON/OFF	ON/OFF
NODES					
IL2RBT	OFF	OFF	ON	ON	OFF
BclxL	ON	ON	OFF	OFF	OFF
IFNGT	ON	ON	ON	ON	OFF
PDGFR	ON	ON	ON	ON	OFF
IFNG	OFF	OFF	OFF	OFF	OFF
GAP	OFF	OFF	OFF	OFF	OFF
Proliferation	OFF	OFF	OFF	OFF	OFF
GZMB	OFF	OFF	ON	ON	OFF
RAS	ON	ON	ON	ON	OFF
TPL2	ON	ON	ON	ON	OFF
FasT	ON	ON	ON	ON	OFF
FLIP	ON	ON	ON	ON	OFF
LCK	ON	ON	ON	ON	OFF
NFAT	ON	ON	ON	ON	OFF
FasL	ON	ON	ON	ON	OFF
Caspase	OFF	OFF	OFF	OFF	OFF
NFKB	ON	ON	ON	ON	OFF
IAP	ON	ON	ON	ON	OFF
BID	OFF	OFF	OFF	OFF	OFF
Cyto. Signal.	OFF	OFF	ON	ON	OFF
TNF	ON	ON	ON	ON	OFF
MCL1	OFF	OFF	ON	ON	OFF
Ceramide	OFF	OFF	OFF	OFF	OFF
GRB2	ON	ON	ON	ON	OFF
PI3K	ON	ON	ON	ON	OFF
SMAD	ON	ON	ON	ON	OFF
P27	OFF	OFF	ON	ON	OFF
ZAP70	ON	ON	OFF	OFF	OFF
CREB	OFF	OFF	OFF	OFF	OFF
DISC	OFF	OFF	OFF	OFF	OFF
IL2RB	OFF	OFF	ON	ON	OFF
Fas	OFF	OFF	OFF	OFF	OFF
IL2RA	Oscillates	Oscillates	OFF	OFF	OFF
S1P	ON	ON	ON	ON	OFF
ERK	ON	ON	ON	ON	OFF
SPHK1	ON	ON	ON	ON	OFF
A20	ON	ON	ON	ON	OFF
MEK	ON	ON	ON	ON	OFF
CTLA4	OFF	OFF	OFF	OFF	OFF
TBET	OFF	OFF	ON	ON	OFF
RANTES	ON	ON	ON	ON	OFF
SOCS	OFF	OFF	OFF	OFF	OFF
sFas	ON	ON	ON	ON	OFF
IL2RAT	ON	ON	OFF	OFF	OFF
TCR	OFF	OFF	OFF	OFF	OFF
STAT3	OFF	OFF	ON	ON	OFF
GPCR	ON	ON	ON	ON	OFF
P2	ON|OFF	OFF	ON|OFF	OFF	OFF
TRADD	OFF	OFF	OFF	OFF	OFF
PLCG1	ON	ON	ON	ON	OFF
FYN	OFF	OFF	ON	ON	OFF
IL2	ON	ON	OFF	OFF	OFF
JAK	OFF	OFF	ON	ON	OFF
Apoptosis	OFF	OFF	OFF	OFF	ON

Table 2: The attractors of T-LGL leukemia survival network. This table shows the state of the nodes for all possible combinations of input signals without the presence of antigen (Stimuli=OFF).

In this research, a detailed analysis of the T-LGL signaling network is found by predicting the fuzzy membership of all possible initial states to understand the long-term behavior of the underlying disease. An implementation of an asynchronous Boolean dynamic framework will be used to verify the attractors of the system and analyze their basins of attraction. The analysis of initial states of the Boolean network will allow for confirmation or prediction of fuzzy membership. This will help identify structure or find fuzzy membership clusters which will be used to find states that lead to healthy attractors.

Justification of the Literature and Identification of Prior Work

Large realistic networks are difficult to simulate and analyze due to computational intractability. Several current studies implemented a network reduction approach that consists of iteratively removing single nodes that do not regulate their own function and simplifying the redundant transfer functions using Boolean algebra (Zanudo & Albert, 2013). By pinpointing and eliminating the stabilized nodes, and iteratively removing a node that has one incoming edge and one outgoing edge while connecting its input $node_i$ to its target $node_j$, a large network can be greatly simplified, which could be more manageable in simulation and analysis. The main advantage of this reduction method is that very large networks, with 200 or more nodes, can be simulated with ease (Zanudo & Albert, 2013). The reduced network can be used to infer properties about the original network to better understand the role and dynamics of its network topology.

Although this simplification can increase performance for attractor search and GRN simulation, this method cannot provide fuzzy membership analysis for individual states. For instance, with the T-LGL network, when a start state s transitions into attractors through multiple samplings, fuzzy membership analysis is not necessarily the same with samplings of P2 being *on*

compared to samplings of P2 being *off*. Thus, this research will focus on asynchronous Boolean networks with sampling of a subset of initial start states, overlapping attractors within a full realistic network, fuzzy membership analysis, and machine learning methods to identify rules to predict the outcome of other states within a network. This fine grain collection of data for individual states will help identify fuzzy membership clusters to determine whether a state transitions into a healthy or unhealthy attractor using machine learning methods.

Identification of Gaps in the Literature

This research will investigate fuzzy membership vectors of large state spaces in asynchronous Boolean networks. The size and complexity of realistic GRNs make it infeasible to estimate fuzzy membership vectors for all states. Network reduction is a method used to bypass this limitation (Saadatpour et al., 2011). Of particular interest is the use of fuzzy sets to characterize a state's degree of membership in different basins of attraction, based on the likelihood of transitioning from the state to a given attractor. The three heuristics that are used to estimate fuzzy membership degree in this research are decision trees, SVMs, and naïve Bayesian classifiers. This work will also be applied to other GRNs such as ABA, Immune Bb, Cardiac Development, and Mammalian Cell Cycle to validate the approach.

Modeling and simulation will determine fuzzy membership vectors for a select sample of states, and the use of machine learning will be used to identify patterns in membership that would extend to a much larger range of states. The data for which the patterns exist is the collection from Markov Chain Monte Carlo sampling. The discovery of rules for these patterns will yield fuzzy membership vectors for states not subject to simulation. These vectors will make it possible to predict the outcome of states that were not part of the initial sampling.

Analysis of Research Methods

Markov Chain Monte Carlo method is a technique that solves the problem of sampling from a complicated distribution such as asynchronous Boolean networks. It is a technique for estimating by simulation of the expectation of a statistic in a complex model. Successive random selections form a Markov chain, the stationary distribution of which is the target distribution (Gilks, 2005). It is particularly useful for the evaluation of posterior distributions in asynchronous Boolean networks and complex Bayesian models.

For this research, the definition of the sampling problem is written to support asynchronous Boolean networks. Let S be a distribution over a finite set of states $\{s_1, \dots, s_k\}$. Given a state $s_i \in S$, the simulation selects at random, with equal likelihood, a successor state s_j based on updating a variable of s_i with the transition function $x_{i^*}(t+1) = f_{i^*}(x_t)$. This random walk on the graph will lead to an attractor. An attractor will be encountered because all asynchronous Boolean networks are absorbing Markov chains (Xiao, 2009). For example, every state s_i in an absorbing Markov chain will transition to another state s_{new} eventually forming cycles with no escape. With the identification of attractors and multiple samplings per state s_i to determine fuzzy membership to those attractors, the state space structure can be identified and machine learning can be used to predict the other states that are not part of the initial sampling.

Monte Carlo methods rely on repeated random sampling of start states to obtain fuzzy membership results. They are a widely used class of computational algorithms for simulating the behavior of various physical and mathematical systems, and for other computations (Hammersley & Handscomb, 1964) such as system biology and financial engineering. Furthermore, the methods also rely on repeated random asynchronous sampling of update functions to transition from one state to another until an absorbing state is reached. Thus, with

multiple samplings of one state, its fuzzy membership can be estimated by the percentage of times it reaches each absorbing state.

A state in a Markov chain is an absorbing state if, once the state is reached, it is impossible to leave. A Markov chain is an absorbing chain if there is at least one absorbing state and it is possible to go from any state to at least one absorbing state in a finite number of steps (Grinstead & Snell, 1997). For Boolean networks, absorbing states of a Markov chain will represent attractors. For this research, attractors that have multiple states forming cycles will be treated as a *super* state to maintain the definition of absorbing Markov chains. Thus, once a Markov chain reaches these *super* states, which are irreducible and a recurrent sets of states, it cannot escape that set. Therefore, in determining the probability of reaching a recurring set of states from the initial state, it will only be necessary to find the probability of reaching that *super* state or recurring set.

Absorbing Markov chains have the property that the powers of the transition matrix approach a limiting matrix (Grinstead & Snell, 1997). For example, for the transition matrix A, there is some integer n_0 such that $A^m = A^n$ for all $m, n >= n_0$. A transition matrix is used to describe the transitions of a Markov chain, and a limiting matrix is used to describe the probability of reaching an absorbing state or attractor. With transition matrices, for Markov chains with one or more absorbing states to have limiting matrices, there is at least one absorbing state, and the possibility must exist to go from each non-absorbing state to at least one absorbing state in a finite number of steps. However, using transition matrices and limiting matrices are very powerful tools for small Boolean networks. For large GRNs, using transition and limiting matrices is not feasible. The size of a matrix for T-LGL would be $2^{60}x2^{60}$, and computing its powers is intractable. Thus, a subset of initial states must be sampled; calculating their fuzzy

membership status into each attractor and using machine learning methods, such as decision trees, SVMs, and naïve Bayesian classifiers, to predict all other states not part of the initial sampling process. These predictions will help researchers test their treatments on states that were not part of the initial sampling without having to send those states through the sampling process, which can take a large amount of time to complete.

A decision tree is a flowchart-like structure consisting of decision nodes, chance nodes, and end nodes. Each internal node represents a *test* on an attribute, with each branching from the nodes representing the outcome of the *test*. Each leaf node represents a decision taken after computing all of the attributes, which is known as a class label. The paths from root to leaf represent classification rules. The goal of decision trees is to learn how to classify objects by analyzing a set of instances of already solved cases whose classes are known (Podgorelec, Kokol, Stiglic, & Rozman, 2002). Learning input consists of a set of attribute-value vectors, each belonging to a known class, and the output consists of a mapping from attribute values to classes (Podgorelec, Kokol, Stiglic, & Rozman, 2002).

For this research, decision trees will be used as a variant of one-against-all formulation. The ith decision function $g_i(x)(i = 1, \dots, n-1)$ is determined, such that if x belongs to class i, then $g_i(x) > 0$, and when x belongs to one of the classes $\{i+1, \dots, n\}$, then $g_i(x) < 0$. Therefore, in classifying x into a class i, starting from $g_1(x)$, the first positive $g_i(x)$ is found and classifies x into class i. If there is no such i among $g_i(x)(i = 1, \dots, n-1)$, then x is classified into class n (Esposito, 2010).

Decision trees are relatively easy to understand and interpret because the inferred classification rules flow from a root node to a leaf node, and this path is based on conditions created from a training set of data. Each leaf node represents one classification rule. Laypeople

are able to understand decision tree models after a brief explanation. Furthermore, decision trees have value even with little hard data. Important insights can be generated based on experts describing the alternatives, probabilities, and costs along with preferences for outcomes. Also, decision trees allow the addition of new possible scenarios, and help determine worst, best, and expected values for different scenarios. However, there are two disadvantages of decision trees. First, information gained in decision trees can be biased in favor of those attributes with more levels (Deng, Runger, & Tuv, 2011) for data that includes categorical variables with different number of levels. The other disadvantage, calculations can get very complex particularly if many values are uncertain or if many outcomes are linked.

Another method for making predictions is Support Vector Machines (SVM). SVMs are kernel-based methods of classification that employ a maximum margin approach using a hyperplane to separate the classes. SVMs are based on the theory of decision planes that describe decision boundaries (Shawe-Taylor & Sun, 2009). For classes that are not linearly separable, a kernel function is used to transform the problem into a higher dimensional space so separation is possible. SVMs use a decision rule $d(S)$ to predict the outcome of future inputs. The important difference between SVMs and other supervised learning classifiers is that SVMs use the optimization of maximum margin to reduce the number of weights that are nonzero to just a few weights that correspond to the important features that matter in deciding the separating line or hyperplane. These nonzero weights correspond to the support vectors because they support the separating hyperplane.

To find the optimal decision solution, the maximum margin must be found. So, let the states $S = \{s_1, \dots, s_k\}$ belong to classification A or not to classification A. In addition, let the associated labels be $y_i \in \{1, -1\}$, where 1 is classification A and -1 is not in classification A. If

the data is linearly separable, then the determination of the decision function is $d(S) = \boldsymbol{w}^T \boldsymbol{S} + b$, where w is the weighted vector, S is the set of states, and b is the bias term. Thus, $\boldsymbol{w}^T \boldsymbol{s_i} + b$ $\{> 0\ for\ y_i = 1, < 0\ for\ y_i = -1\}$ satisfies linearly separable data, which simplifies to $y_i(\boldsymbol{w}^T \boldsymbol{s_i} + b) \geq 1$. Furthermore, the hyperplane $d(S) = \boldsymbol{w}^T \boldsymbol{S} + b = c, for -1 < c < 1$, forms a separating hyperplane that separates s_i. When $c = 0$, the separating hyperplane is in the middle of the two hyperplanes with $c = 1$ and -1. The distance between the separating hyperplane and the training data sample nearest to the hyperplane is called the margin. Therefore, the wider the margin, the more accurate the predictions become.

There are multiple decision functions that can separate linearly separable classes. Some decision functions are more optimal than others. The SVMs uses criteria to look for a decision boundary that has a maximum distance between any data point from two different classes (Shawe-Taylor & Sun, 2009). Therefore, an optimal separating hyperplane must be found. The optimal separating hyperplane can be obtained by solving $minimize\ Q(\boldsymbol{w}, b) = \frac{1}{2}\|w\|^2$ subject to $y_i(\boldsymbol{w}^T \boldsymbol{s_i} + b) \geq 1$. The data that supports $y_i(\boldsymbol{w}^T s_i + b) = 1$ are known as support vectors, and all data that supports $y_i(\boldsymbol{w}^T s_i + b) \geq 1$ are known as feasible solutions. The optimal separating hyperplane and decision function are constrained to their current state space and must be converted into an equivalent dual problem whose number of variables is the number of training data.

The constrained problem must be converted into an unconstrained problem using non-negative Lagrange multipliers. Let $Q(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\boldsymbol{w}^T \boldsymbol{w} - \sum_{\boldsymbol{i=1}}^{\boldsymbol{S}} \alpha_{\boldsymbol{i}}\{y_i(\boldsymbol{w}^T \boldsymbol{s_i} + b) - 1\}$, where $\alpha = \{\alpha_i, \ldots, \alpha_K\}$ are the nonnegative Lagrange multipliers. Using this formula, the KKT conditions, $\frac{\partial Q(\boldsymbol{w},b,\alpha)}{\partial \boldsymbol{w}} = 0$ and $\frac{\partial Q(\boldsymbol{w},b,\alpha)}{\partial b} = 0$, can be reduced to $w = \sum_{i=1}^{S} \alpha_{\boldsymbol{i}} y_i \boldsymbol{s_i}$ and

$\sum_{i=1}^{S} \alpha_i y_i \boldsymbol{s_i} = 0$, respectively. When substituted back into the original unconstrained problem, the following dual problem is obtained: $maximize\ Q(\alpha) = \sum_{i=1}^{S} \alpha_{\boldsymbol{i}} - \frac{1}{2}\sum_{i,j=1}^{S} \alpha_i \alpha_j y_i y_j s_i^T s_j$. Thus, this reduces to an update decision rule of $d(s) = \sum \alpha_i y_i s_i^T s_{unknown} + b$. The formula works for linearly separable datasets. However, datasets that are not linearly separable need an additional step.

When datasets are not linearly separable then another perspective in the state space has to be taken. Transforming to a higher dimension within the state space will lead to a separable dataset (Shawe-Taylor & Sun, 2009). The decision rule, $d(s) = \sum \alpha_i y_i s_i^T s_{unknown} + b$, can have s_i^T and $s_{unknown}$transformed by a kernel function. One such kernel function that satisfies this is $k(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$, which is why SVMs are known as kernel methods. This has led to SVMs excelling in realms where other machine learning methods are dominant, and many believe that SVMs are the best *off-the-shelf* supervised learning algorithms in the market (Ng, 2015).

Some advantages of SVMs are maximization of generalization ability, no local minima, applicable to a wide range of applications, and robustness to outliers. Some disadvantages are extension to multiclass problems, long training time, and the time and effort needed for the selection of parameters. However, by introducing a kernel function, the SVMs will gain flexibility in separating classes that are not linearly separable.

An easier method to implement for predictions is naïve Bayesian classifiers. These classifiers are based on Bayes' Theorem with statistically independent assumptions between predictors. *Naïve* refers to the assumption that data attributes are independent, and the Bayesian method still can be optimal even when this attribute independency is violated (Domingos & Pazzani, 1997). A naïve Bayesian model is easy to build, with no complicated iterative parameter

estimation, making it particularly useful for very large networks. Despite its simplicity, the naïve Bayesian classifier often does surprisingly well and is widely used because it frequently outperforms more sophisticated classification methods. The naïve Bayesian classifier is easy to understand, explain, and debug, and could be modified with new training data without having to rebuild it.

Bayes' Theorem provides a way to calculate the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$, and $P(x|c)$, such that $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$, where c is the class, and x is the data, predictor, or attributes. $P(c|x)$ is the *posterior probability distribution* for a classification given its attributes, $P(c)$ is the *prior probability distribution* for a classification, $P(x|c)$ is the *likelihood function* which is the probability of attributes given the classification, and $P(x)$ is the *prior probability distribution* for attributes. A naïve Bayesian classifier assumes that the effect of the value of a predictor on a given class is independent of the values of the other predictors. This assumption is called class conditional independence. Furthermore, with knowledge of the probabilities for each attribute, Bayes' Theorem can be reduced to $P(c|x_1, \dots, x_n) = P(c)\prod_{i=1}^{n} P(x_i|c)$. This rule is extremely fast at calculating the probability of being in a class compared to other sophisticated algorithms.

For example, calculating the posterior probability can be done by constructing a frequency table for each attribute against the classification target, then transforming the frequency tables to likelihood tables, and finally using the naïve Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of the prediction indicated by $d = \operatorname{argmax}_{k\in\{1,\dots,K\}} P(c_k)\prod_{i=1}^{n} P(x_i|c_K)$, where d is the decision function for the naïve Bayesian classifier. An advantage of using a naïve Bayesian classifier is that it only requires a small amount of training data to estimate the parameters

necessary for classification. This allows a small proportion of states from a Boolean network to be sampled to accurately predict all of the other states not part of the initial sampling.

Summary

One particular use of synchronous and asynchronous Boolean networks is the modeling and simulation of GRNs. Synchronous Boolean networks updates to functions are applied simultaneously at each discrete time point, whereas, asynchronous Boolean networks update one function, chosen randomly, at each discrete time point. However, with asynchronous Boolean networks, an initial state in one sampling can transition into an attractor, whereas in another sampling, an initial state can transition into a different attractor. This means that an initial state can have overlapping attractors, the set of attractors reached by some sequence of state transitions. Using Boolean networks can help to understand how GRNs behave and input nodes can alter that behavior. For instance, understanding how genes work together to comprise functional cell and life to cope with environments and disease will provide researchers with novel drug targets, sensitive diagnostics for individualized therapy, and ways to manipulate its topology to cure disease. In this research, the T-LGL GRN is used, which is a large network with 60 nodes, including six input nodes. In previous studies, the attractors have been identified, and this study has validated their existence. Furthermore, large networks are difficult to simulate, so the implementation of network reduction was introduced to make simulation much more manageable. This method fails to find fuzzy membership status of individual states. As a result, this research will use machine learning to predict the fuzzy membership outcome of states. The techniques that this study will use to achieve its goals are Markov Chain Monte Carlo methods, SVNs, decision trees, and naive Bayesian networks.

Chapter 3

Methodology

Introduction

Due to the intractable nature of identifying fuzzy membership for every state in a large GRN, machine learning techniques will be used to predict the basin of attractions to which these states belong. The methodology for this research is to use the T-LGL GRN model and machine learning methods as inputs, and $<\tau_{low}, \tau_{high}>$ thresholds as the parameters for the output classes. Next, collecting data from executing simulations starting from random states provides the training and testing dataset for machine learning methods. Additionally, an exploratory data analysis, to include visualizations and histograms, can expose the structure of fuzzy membership to attractors. The output class or rule sets produced from the machine learning methods will be tested to predict accurately fuzzy membership of other states at random. Once the process is finalized, it will be applied to other GRNs to validate its accuracy.

Hypotheses

Realistic GRNs, such as T-LGL, possess membership structures for its states based on clusters defined by range $\tau = [\tau_{low,}\tau_{high}]$. A function $F: S \rightarrow [0,1]^n$ maps a set of states to their fuzzy membership vectors, where there are n attractors. Then F_A projects to the component for attractor A. In other words $F_A(s)$ is the component of the fuzzy membership vector $F(s)$ corresponding to attractor A. Thus, there exist a fuzzy membership vector function $F_A(s)$ for attractor A, such that there are a set of states whose values lie within a threshold range τ, which produces the output class $\langle A, \tau \rangle$. Furthermore, there also exist a fuzzy membership function $F_A(s)$, such that there are a set of different states whose values *do not* lie within a threshold range

τ, which produces the output class $\neg\langle A, \tau\rangle$. Also, executing $N = 1000$ simulations on state s yields an approximation to $F(s)$. Additionally, for a state s, its sets are defined as $\langle A, \tau\rangle = \{s \mid F_A(s) \in \tau\}$ and $\neg\langle A, \tau\rangle = \{s \mid F_A(s) \notin \tau\}$. Furthermore, state s is labeled by $\langle A, \tau\rangle$ if and only if $F_A(s) \in \tau$; else s is labeled by $\neg\langle A, \tau\rangle$.

For this study, healthy cluster of states are defined within a probability threshold range of $\tau = [0.9, 1.0]$. Additionally, a healthy state is a state that can reach apoptosis more than 90% of the time. A state that lies in an overlapping basin of attraction of multiple attractors will have fuzzy membership in each of those attractors. Through simulation, a state s in an overlapping basin will transition into one of the attractors. Furthermore, with a large number of samplings, a state s will eventually transition into all attractors that are part of the overlapping basin of attraction. The number of times a state s transitions into a specific attractor A determines its fuzzy membership with that attractor. For example, let a state s transition into two overlapping attractors which are attractors A and B. This represents one sampling. There will be *N*=1000 samplings. During each sampling, state s will transition to either attractor A or attractor B. Once the sampling phase is completed, state s will have transitioned to attractor A x_1 times and attractor B x_2 times, where $x_1 + x_2 = N$ hits. Therefore, a state's fuzzy membership in each attractor when added together will equal one $[\frac{x_1}{1000} + \frac{x_2}{1000} = 1]$ and the list of fuzziness for each attractor is the state's fuzzy membership vector. This generalizes over any number of attractors, e.g. $[\frac{x_1}{1000} + \frac{x_2}{1000} \ldots \frac{x_n}{1000} = 1]$. Once the fuzzy membership is known for a group of states within attractor A, then the discovery of good output classes bound by τ_{low} and τ_{high}could be determined for that attractor. For this study, $\tau = [\tau_{low} = 0.9, \tau_{high} = 1.0]$ was chosen to form a cluster states with a high success rate of reaching attractor A. Finally, for all states that have

similar fuzzy membership to the same attractor, the data collected can be used to train machine learning methods to predict other states that were not initially part of the simulation.

States form natural clusters in the vector space N^A where A is the number of attractors. Clusters can be discerned using such methods, such as k-means, etc., and can also be visualized such as the one in Figure 6. For example, a cluster could be defined by $\tau = [\tau_{low,}\tau_{high}] = [0.9, 1.0]$. However, this may not be representative to all GRNs. $\tau = [\tau_{low,}\tau_{high}]$ has to be determined independently for each GRN. If visualizations are not enough to form clusters then the usage of clustering methods is another possibility.

Machine learning methods when trained by clustering data defined by $\tau = [\tau_{low,}\tau_{high}]$ can identify accurate fuzzy membership vectors for states. With data collected from a simulation, a classifier is trained to determine whether a state s_i, not part of the original simulation, is in the range of τ or is outside the range of τ for binary classification. This means that a group of states can transition into attractor A within the range $\tau = [0.9, 1.0]$. Furthermore, all other states can transition into attractor A with less probability within range $\tau = [0.0, 0.9]$. Therefore, a trained classifier with a state s as its input can accurately determine if state s is either in the range of τ or outside the range of τ. However, different classifiers may have different variations of accuracy in determining their output.

The comparison of machine learning methods' effectiveness for predicting fuzzy membership vectors for states of a GRN are similar. For instance, based on the visualization in Figure 6, a cluster is defined by $\tau = [\tau_{low}, \tau_{high}]$, and a classifier such as a decision tree has prediction accuracy of 97% from a previous experiment. In addition, machine learning methods such as Naive Bayesian Classifiers and SVMs also had similar accuracy of 97%. The feature set of a GRN, such as T-LGL, will have a subset of features dominating the outcome of the

classifiers output. Therefore, future experiments are expected to have similar prediction accuracy.

The most useful output classes for this application of machine learning methods is: given attractor A and a specified threshold range τ, the output class of a state is positive if the probability of reaching attractor A of a state lies within the specified range τ, else it is negative if the probability of reaching attractor A lies outside the specified range τ. This would allow a clinical biologist to focus on healthy and unhealthy attractors. Furthermore, a clinical biologist could implement treatments that can improve the chances of a state s, defined by a specified range τ, to transition into a healthy attractor A.

Research Design

This research uses statistical machine learning to predict the output class of a state, where that state was not part of the original simulation. The research design has two inputs, a GRN and a classifier, and one output class. The output class consist of $<A, \tau>$ and $\neg<A, \tau>$ where A is the attractor and τ is the threshold range. A state s classifies to attractor A with probability $P(s, A) \in \tau$, otherwise it classifies to $\neg A$.

There will be a total of eight experiments, four for T-LGL, and one for each of the following GRNs: ABA, cardiac development, immune Bb, and mammalian cell cycle. Some GRNs will have source input nodes that will remain static throughout the sampling process of the experiment. For example, T-LGL has three nodes, stimuli (the presence of cancer), stimuli2 (the extreme presence of cancer), and CD45. The GRN must have these nodes remain in its static state to represent real world tests. Each experiment will follow a research design schematic to collect data from sampling to train classifiers for predictions.

Figure 7 is the research design schematic for this study in which the implementation is an experiment. There are two sets of inputs, a GRN and a classifier. The GRN may have input nodes that remain static throughout the experiment, such as the T-LGL GRN. For T-LGL, there will be four set of experiments. The difference between these experiments are the source input nodes will start with different values that will remain static throughout the experiment. A subset of states will be selected randomly for sampling to collect data which will determine fuzzy membership vectors for each of those states.

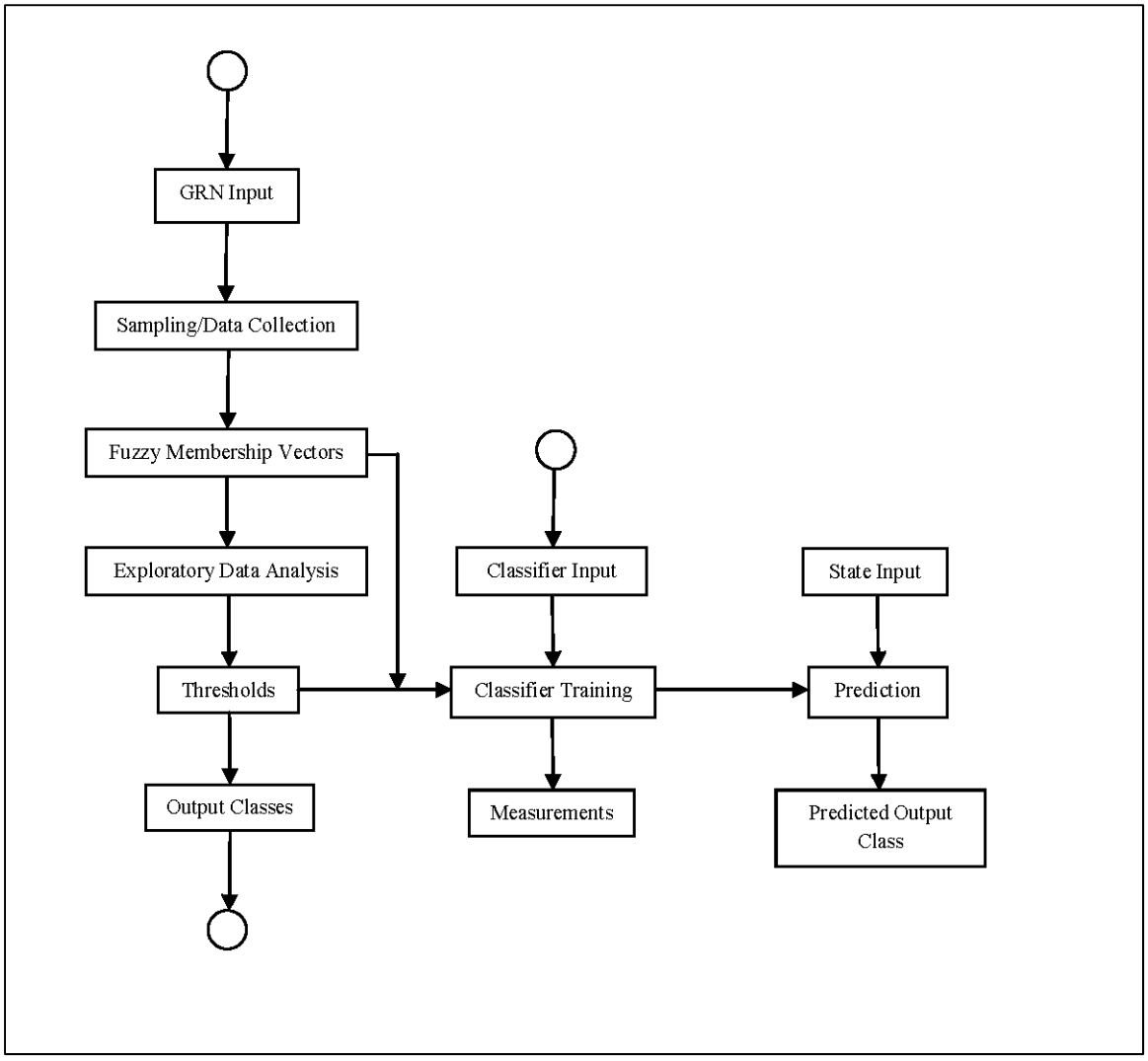

Figure 7: Research Design Schematic

There will be N=1000 number of samplings of each state s_i to determine its fuzzy membership vector. During the sampling process, each state involves a stochastic process of running through asynchronous transitions to reach an attractor. For each state s_i, N=1000 samplings will be taken, and each sampling will transition into an attractor. Once the sampling process is complete, each state s_i fuzzy membership vector will be determined. Once a list of fuzzy membership vectors has been compiled, exploratory data analysis can be performed to determine the output classes $<A, \tau>$ and $\neg<A, \tau>$.

The exploratory data analysis step of this research design is used to form clusters of states with similar fuzzy membership vectors, and the purpose is to devise significant output classes defined by $\langle A, \tau \rangle$. Visualizations, such as Figure 6 discussed in Chapter 1, can be used to identify clusters. Once the clusters have been identified, each cluster is assigned a threshold range and placed into a list. The list of threshold ranges for each attractor represents the output classes.

For a state that was not part of the initial simulation, the prediction of its output class will be determined by a classifier trained with the fuzzy membership vectors and the threshold ranges. The data will randomly be partitioned into bins using k-fold cross-validation. The purpose of cross-validation is to define a dataset or determine the widest margin to test the classifier in the training phase, which will prevent the overfitting problem, and give insight on how the classifier will generalize to an independent dataset. The goal of the k-fold cross-validation is to *measure* how accurate the predictions are for a given classifier.

Variables

For much of this study, the focus will be on large realistic networks. Each experiment comprises a GRN, which is executed with each of the three machine learning algorithms, and a set of output class as another variable. The GRN network is one of the inputs. Another set of

inputs, the choice of machine learning algorithms, will be comprised of decision trees, naïve Bayesian classifiers, or SVNs. These classifiers will undergo training and testing to accurately predict fuzzy membership of a random state. Particular values of tau are GRN-dependent and will be determined by exploratory data analysis. Another might indicate if a state has specified fuzzy membership in *any* attractor. For instance, a state *s* that is in the output class $\neg\langle A, \tau\rangle$ may still have fuzzy membership in attractor *A*, but such membership is outside of the τ range.

Choice of Genetic Regulatory Network

As previously mentioned, the T-LGL GRN is the network of choice for this research. While this research will focus on T-LGL for presentation and refinement of the methods used, it also will present results for other networks such as Abscisic Acid signaling model (ABA), Mammalian Immune response to B. Bronchiseptica infection (Immune Bb), Cardiac Development, and Mammalian Cell Cycle. T-LGL is a large realistic GRN model created from hundreds of literature sources and validated experimentally (Zhang et al., 2008). Due to this validation, this network makes an ideal situation to apply this approach, which is purely computational and requires no expert level knowledge of the disease system. If the results on this network show accurate predictions, then the approach may prove useful in predicting other large networks, for which there is a lack of expert-level knowledge and/or the ability to simplify systematically. Figures 8 and 9 show the structure of the T-LGL GRN and Boolean network respectively.

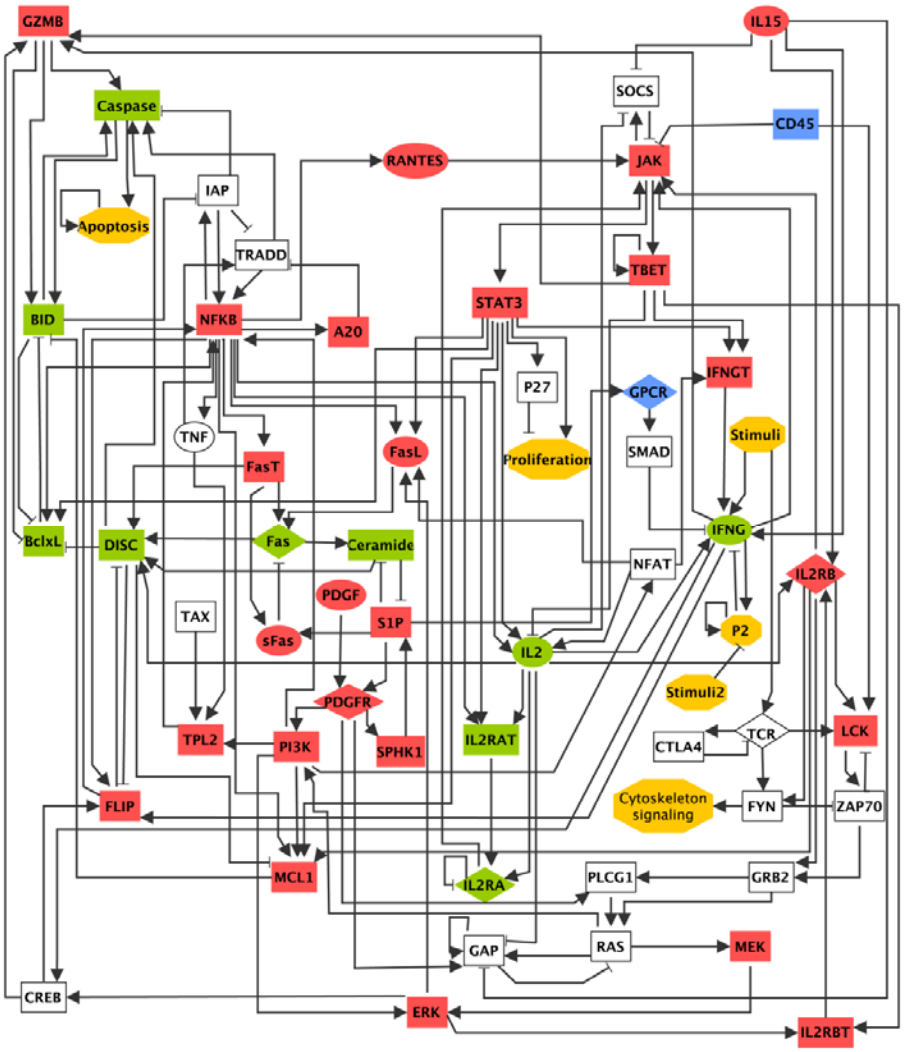

Figure 8: The T-LGL survival signaling GRN. The rectangles, ellipses, and diamonds indicate intracellular components, extracellular components, and receptors. In addition, the knowledge of leukemic cells is represented by red, green, blue, and white, which indicates highly active components, inhibition, deregulated, and unknown states. An arrowhead at the end of an edge indicates activation, and a short perpendicular bar at the end of an edge indicates inhibition. The inhibitory edges from Apoptosis to other nodes are not shown (Zhang et al., 2008).

T-LGL Boolean Network

```
IL2RBT     *= (ERK ∧ TBET) ∧ ¬Apoptosis
BclxL      *= (NFKB ∨ STAT3) ∧ ¬(BID ∨ GZMB ∨ DISC ∨ Apoptosis)
IFNGT      *= (TBET ∨ STAT3 ∨ NFAT) ∧ ¬Apoptosis
PDGFR      *= (S1P ∨ PDGF) ∧ ¬Apoptosis
IFNG       *= ((IL2 ∨ IL15 ∨ Stimuli) ∧ IFNGT) ∧ ¬(SMAD ∨ P2 ∨ Apoptosis)
GAP        *= (RAS ∨ (PDGFR ∧ GAP)) ∧ ¬(IL15 ∨ IL2 ∨ Apoptosis)
Proliferation *= STAT3 ∧ ¬(P27 ∨ Apoptosis)
GZMB       *= ((CREB ∧ IFNG) ∨ TBET) ∧ ¬Apoptosis
RAS        *= (GRB2 ∨ PLCG1) ∧ ¬(GAP ∨ Apoptosis)
TPL2       *= (TAX ∨ (PI3K ∧ TNF)) ∧ ¬Apoptosis
FasT       *= NFKB ∧ ¬Apoptosis
FLIP       *= (NFKB ∨ (CREB ∧ IFNG)) ∧ ¬(DISC ∨ Apoptosis)
LCK        *= (CD45 ∨ ((TCR ∨ IL2RB) ∧ ¬ZAP70)) ∧ ¬Apoptosis
NFAT       *= PI3K ∧ ¬Apoptosis
FasL       *= (STAT3 ∨ NFKB ∨ NFAT ∨ ERK) ∧ ¬Apoptosis
Caspase    *= ((((TRADD ∨ GZMB) ∧ BID) ∧ ¬IAP) ∨ DISC) ∧ ¬Apoptosis
NFKB       *= ((TPL2 ∨ PI3K) ∨ (FLIP ∧ TRADD ∧ IAP)) ∧ ¬Apoptosis
IAP        *= NFKB ∧ ¬(BID ∨ Apoptosis)
BID        *= (Caspase ∨ GZMB) ∧ ¬(BclxL ∨ MCL1 ∨ Apoptosis)
Cytoskeleton signaling *= FYN ∧ ¬Apoptosis
TNF        *= NFKB ∧ ¬Apoptosis
MCL1       *= (IL2RB ∧ STAT3 ∧ NFKB ∧ PI3K) ∧ ¬(DISC ∨ Apoptosis)
Ceramide   *= Fas ∧ ¬(S1P ∨ Apoptosis)
GRB2       *= (IL2RB ∨ ZAP70) ∧ ¬Apoptosis
PI3K       *= (PDGFR ∨ RAS) ∧ ¬Apoptosis
SMAD       *= GPCR ∧ ¬Apoptosis
P27        *= STAT3 ∧ ¬Apoptosis
ZAP70      *= LCK ∧ ¬(FYN ∨ Apoptosis)
CREB       *= (ERK ∧ IFNG) ∧ ¬Apoptosis
DISC       *= (FasT ∧ ((Fas ∧ IL2) ∨ Ceramide ∨ (Fas ∧ ¬FLIP))) ∧ ¬Apoptosis
IL2RB      *= (IL2RBT ∧ (IL2 ∨ IL15)) ∧ ¬Apoptosis
Fas        *= (FasT ∧ FasL) ∧ ¬(sFas ∨ Apoptosis)
IL2RA      *= (IL2 ∧ IL2RAT) ∧ ¬(IL2RA ∨ Apoptosis)
S1P        *= SPHK1 ∧ ¬(Ceramide ∨ Apoptosis)
ERK        *= (MEK ∧ PI3K) ∧ ¬Apoptosis
SPHK1      *= PDGFR ∧ ¬Apoptosis
A20        *= NFKB ∧ ¬Apoptosis
MEK        *= RAS ∧ ¬Apoptosis
CTLA4      *= TCR ∧ ¬Apoptosis
TBET       *= (JAK ∨ TBET) ∧ ¬Apoptosis
RANTES     *= NFKB ∧ ¬Apoptosis
SOCS       *= JAK ∧ ¬(IL2 ∨ IL15 ∨ Apoptosis)
sFas       *= FasT ∧ S1P ∧ ¬Apoptosis
IL2RAT     *= (IL2 ∧ (STAT3 ∨ NFKB)) ∧ ¬Apoptosis
TCR        *= Stimuli ∧ ¬(CTLA4 ∨ Apoptosis)
STAT3      *= JAK ∧ ¬Apoptosis
GPCR       *= S1P ∧ ¬Apoptosis
P2         *= (IFNG ∨ P2) ∧ ¬(Stimuli2 ∨ Apoptosis)
TRADD      *= TNF ∧ ¬(IAP ∨ A20 ∨ Apoptosis)
PLCG1      *= (GRB2 ∨ PDGFR) ∧ ¬Apoptosis
FYN        *= (TCR ∨ IL2RB) ∧ ¬Apoptosis
IL2        *= (NFKB ∨ STAT3 ∨ NFAT) ∧ ¬(TBET ∨ Apoptosis)
JAK        *= (IL2RA ∨ IL2RB ∨ RANTES ∨ IFNG) ∧ ¬(SOCS ∨ CD45 ∨ Apoptosis)
Apoptosis *= Caspase ∨ Apoptosis
```

Figure 9: T-LGL signaling Boolean Network (Zhang et al., 2008).

Network Analysis

The network analysis for this research will include running simulations from random start states and using methods such as visualization tools and clustering to discover fuzzy membership structure. Implementing the asynchronous search algorithm, the *ForwardSet()* and

ValidateAttractor() functions will assist with the simulations. Sampling will collect data based on the Markov Chain Monte Carlo method. The process starts from a randomly chosen set of states $S = \{s_1, \dots, s_k\}$, where k represents the number of states in a subset of the state space. In Table 3, $k=2^n$, where n=15 in all experiments except for experiment VIII (mammalian cell cycle) where n=20. The mammalian cell cycle is a small GRN which contains a total of 20 nodes, therefore all nodes were used. Each state is sampled N=1000 times. Then, a high number of state transitions are performed in each sample to reach a potential attractor, where the maximum number of transitions is r=5000. For this study, r=5000 is sufficient for all states to reach an attractor. No statistics were collected for the number of iterations sufficient to reach an attractor. The algorithm will use the *ForwardSet()* function to validate the attractor. The attractor is placed into a list and it will be used for validation during the sampling process. During the sampling process, a state s_i will transition to another state using the *T_Asynch()* function, and the new state s_j will be validated against the list of attractors. If the new state s_j is not in the list of attractors, then more transitions will take place until an attractor is hit; this hit will count as one sample. A large number of samples, N=1000, will be taken for each state $s_i \in \{s_1, \dots, s_k\}$, because each state can potentially hit different attractors per sample.

Experiments	2^n (number of states sampled)	N (number of simulations per sampled state)	r (number of transitions allowed per state)
Experiment I - LGL	n=15, 32768 states	N=1000	r=5000
Experiment II - LGL	n=15, 32768 states	N=1000	r=5000
Experiment III - LGL	n=15, 32768 states	N=1000	r=5000
Experiment IV - LGL	n=15, 32768 states	N=1000	r=5000
Experiment V - ABA	n=15, 32768 states	N=1000	r=5000
Experiment VI - Cardiac Development	n=15, 32768 states	N=1000	r=5000
Experiment VII - Immune	n=15, 32768 states	N=1000	r=5000
Experiment VIII - Mammalian Cell Cycle	n=20, 1048576 states	N=1000	r=5000

Table 3: Variables used for the experiments.

Formulate Criteria

Next, all of the samples collected were saved to a file so other applications can access the data quickly and reliably during subsequent exploratory data analysis. Applications that can use

this data are visualization tools such as Tableau. The purpose of exploratory data analysis for this study is to identify patterns from the data collected. Two approaches were taken to gain intuition into the data.

The first approach was to identify a GRN independent output class. For instance, an output class for T-LGL is defining the threshold to be $\tau = [\tau_{low,}\tau_{high}] = [0.9, 1.0]$. Attractor A is representing the healthy option, apoptosis, which is the death of a cancerous cell. Thus, all states would be classified as attractor A, apoptosis, or not attractor A, which means the cancerous cell continues to live and reproduce.

Another approach is to identify a GRN dependent output class. For example, using network analysis mentioned previously can be used as the sampling phase for each state. The number of hits is recorded each time state *s* reaches attractor *A*. Next, a histogram chart is created, similar to Figure 6, starting with states with the lowest number of hits and progressing with states that have larger number of hits. The histogram's hills and valleys is one way of identifying clusters. A cluster is a subset of states with similar fuzzy membership vectors. Once the clusters have been discovered, then rules can be used to predict that a state *s* is leading to attractor A with probability $P(s, A) \in \tau = [\tau_{low,}\tau_{high}]$, otherwise not. τ_{low} and τ_{high} is used to define the lower and upper bounds of a specific cluster. Trying to identify the best clusters for this research is out of scope. Thus, all experiments uses $\tau = [\tau_{low,}\tau_{high}] = [0.9, 1.0]$.

These machine learning methods will help predict fuzzy membership vectors for randomly chosen states. This research is intended to discover rules to predict that a state s leading to attractor A when $\tau = [\tau_{low,}\tau_{high}]$. Once the output classes have been identified, the use of decision trees, SVMs, and naïve Bayesian classifiers will be used in the classification process.

Classification of States

Providing prediction based on statistical machine learning tools, such as decision trees, SVMs, and naïve Bayesian classifiers, will show if a state belongs to a healthy output class. A healthy output class is a class that is most desirable to clinical biologist. For example, reaching apoptosis is the most desirable attractor in T-LGL because this is destroying cancer. Finding states that can reach apoptosis within threshold range $\tau = [\tau_{low,}\tau_{high}]$ is a desirable output class for T-LGL. Once a set of thresholds, τ_{low} and τ_{high}, have been identified, then decision trees will be used to identify a set of rules to predict accurately whether a state s_i belongs to the output class. Cross validation methods will be used to setup training and test sets from collected data in order that the classifiers can be fitted correctly to make accurate predictions. A typical approach in 10-fold cross-validation is to remove the test cases and partition the remaining observations into 10 equal sets. In each of 10 training runs the classifier model is trained on 9 of the 10 sets and the parameters are fine-tuned using the remaining set to improve performance. The model with the fine-tuned parameters is tested using the held-back test data. But, for this research, no test cases will be in usage. The cross-validation process will fine tune the parameters only. SVMs and naïve Bayesian classifiers can be used to compare prediction accuracy. These rules will be used to classify new initial states $s_i \notin S = \{s_1, \dots, s_k\}$ as positive or negative, where S is not part of the initial sampling process. The research will communicate the results through visualization, stories, and/or interpretable summaries.

Summary

Machine learning techniques will be used to predict a state's output class of a given GRN. This research methodology will have a choice of GRN and a choice of a machine learning algorithm, and an output class which is the threshold range and attractor A as the output class.

Data will be collected during the simulation of a GRN on a random set of states. The data collection will be used to identify threshold ranges and to train classifiers to accurately predict a state's output class. The output class or rule sets produced from the machine learning methods will be tested to predict accurate fuzzy membership vectors of other states chosen at random. This methodology will be applied to other GRNs to validate its accuracy. The output class consists an attractor A and a threshold range $\tau = [\tau_{low,}\tau_{high}]$, and the threshold is measured in percentage for fuzzy membership of a random state. While this research will focus on T-LGL for presentation and refinement of the methods used, it also will present results for other networks such as Abscisic Acid signaling model (ABA), Mammalian Immune response to B. bronchiseptica infection (Immune Bb), Cardiac Development, and Mammalian Cell Cycle.

Chapter 4

Results

Introduction

Fuzzy membership vectors are difficult to estimate for all states of large realistic GRNs. A GRN has elements expressed as *on (true)* or *off (false)*, and if it has size n number of elements, it will have a total number of 2^n possible states. Such difficulty can be reduced by taking a subset of states and finding their fuzzy membership vectors. These fuzzy membership vectors can be used to train classifiers to predict the outcomes for the rest of the states with high accuracy.

Finding attractors and fuzzy membership vectors for a subset of randomly chosen states helped to determine the accuracy of fuzzy membership vectors for the rest of the states in realistic GRNs. Attractors were discovered by using randomly chosen states that transitioned into other states using Markov Chain Monte Carlo methods. This approach was used for all the GRNs in this work except for Cardiac Development and Mammalian Cell Cycle, in which all states were chosen for both GRNs. The Cardiac Development GRN has 2^{15}=32768 possible states and Mammalian Cell Cycle has 2^{20}=1048576 possible states. These two GRNs were small enough to choose all states for the study.

After *r=5000* number of transitions, the final resting state is tested to determine if it resides in an attractor. In preliminary tests, where r=2000 and r=4000, sometimes no attractor was reached for some states and the tests had to be repeated until *r=5000* satisfied the problem for all states reaching an attractor. Each state is sampled *N=1000* to estimate its fuzzy membership vector. Next, a new set of randomly selected states are chosen to determine their fuzzy membership vector. Once a threshold range $\tau = [0.9,1.0]$ is established and an attractor A is identified, the output class $<A, \tau>$ is established. The states' fuzzy vectors are used to label

each state; these states now comprise the training set. Once trained, the classifier could predict whether a state can reach the specified attractor or not. The 10-fold cross validation method was used to get an estimate of classifier performance of SVMs, Bayesian classifiers, and decision trees.

The results from exploratory data analysis and predictive analytics are used to answer the following research questions. Could machine learning methods identify accurate fuzzy membership vectors for states within GRNs? Which machine learning methods were most effective in accurately predicting fuzzy membership vectors for states? And, given a threshold, what are the most useful output classes in determining whether a state reaches or not reach an attractor? To accurately answer these questions, cross validation methods were used to train classifiers. While cross validation methods helped classifiers perform better, naive Bayesian classifiers needed principal component analysis for additional preprocessing to be competitive with the other classifiers.

Principal component analysis (PCA) is a statistical method for transforming large number of features for a data set with high correlation into a new set of uncorrelated features referred to as principle components. The state space for the data is reduced to a smaller space while retaining a large amount of variability to prevent the degradation of prediction accuracy of Naive Bayesian classifiers. GRNs, such as T-LGL, have large dimensional data that is highly correlated, which can cause problems for naive Bayes methods. This research uses PCA to improve the performance of Naive Bayesian classifiers to manage such high dimensional data. The performance of SVM and decision trees in this research degraded when using PCA, thus the results do not reflect the usage of PCA with SVMs or decision trees. After using PCA to improve naive Bayesian classifiers, a measure of the importance of the study had to be implemented.

For this research, the significance level was set to 0.01. The statistical significance was attained whenever the observed p-value of a test statistic was less than the significance level defined for this study. The p-value, which was derived from the chi-square value, was the probability of obtaining results at least as extreme as those observed. Also, a confusion matrix (Kohavi, 1998) was used to get information about actual and predicted classifications done by the classification systems. In this study, the confusion matrix was used to determine the chi-square value which leads to computing the p-value.

Experiments I, II, III, and IV

The T-LGL model contains six external input nodes (or source input nodes) which represent extracellular stimuli (Figure 10), adding a stochastic component to the network, or a *background noise* that exists in all biological systems. The nodes are Interleukin 15 (IL15), which is a cytokine that stimulates the proliferation of NK-cells (natural killer cells) and T-cells (Thymus cells); Stimuli, which is an antigen stimulation of leukemia in a T-cell; Platelet-derived growth factor beta polypeptide (PDGF), which is a key master switch in controlling these survival pathways in T-LGL leukemia; Stimuli2, which is a new and stronger antigen stimulation of leukemia in a T-cell; Protein tyrosine phosphatase, receptor type, C (CD45); and Tax p40 Human T-Lymphotropic virus 1 (TAX).

Source Input Nodes

IL15	Interleukin 15
Stimuli	The presence of leukemia
PDGF	Platelet-derived growth factor beta polypeptide
Stimuli2	A stronger presence of leukemia
CD45	Protein tyrosine phosphatase C
TAX	Tax p40 Human T-Lymphotropic virus 1

Figure 10: Six external input nodes for T-LGL experiments.

For each T-LGL experiment, IL15, Stimuli, PDGF, and TAX are expressed (*on*) or set to *true*. Gene expression is the process by which information from a gene is used in the synthesis of

a functional RNA or protein. Stimuli2 and CD45 were chosen to be a combination of *on* or *off*, which produces four experiments. CD45 is a complex enzymatic reactions that is performed by the concerted action of protein kinase and phosphatase. When CD45 is not expressed, the cellular process can result in disease conditions such as leukemia. When Stimilus2 is expressed, this represents a strong presence of leukemia, and when it's not expressed, there is a weak presence of leukemia since stimuli is always expressed. Apoptosis (attractor *A* for this study) was initially set to false (off) for all states. If apoptosis is set to true (*on*), then all states with this setting will reach apoptosis due to this initial setting.

A subset of states, 2^{15} = 32768 states or $\frac{2^{15}}{2^{60}} = 2.8422\ x\ 10^{-12}$% of the states, was chosen randomly. Once chosen, these same states were used in experiments I, II, III, and IV. The number of samplings per state was set to *N*=1000. The number of transitions allowed per state was set to *r*=5000, which for this study was enough for every state to successfully hit an attractor. Initially, *r*=1000 and *r*=2000 were used but there was a high rate of unsuccessful attempts to reach an attractor, whereas *r*=5000 had a 100% success rate of reaching an attractor in all experiments.

Experiment I (Stimuli2=false and CD45=true)

Three attractors were discovered which are shown in Table 1. The healthy attractor *A* (apoptosis) was considered for this study whereas the other two cancerous attractors were treated as unhealthy. The state *s* was considered to be in attractor *A* if and only if it's within the threshold range τ, which forms the output classes of <*A*, τ>. Each state was sampled and the number of hits was recorded to determine its fuzzy membership vector. A histogram chart was created based on these hits in Figure 11.

The output class was used to train classifiers using 10-fold cross-validation to make predictions for other states that were not part of the initial subset of states. A *confusion matrix* of *observed values* was created during the cross-validation step. A *confusion matrix* of *expected values* was also created from the *confusion matrix* of *observed values*. A *chi-square* calculation was derived from both *observed* and *expected confusion matrices*. The *p-value* was calculated from the *chi-square* calculation to show the significance of the experiment.

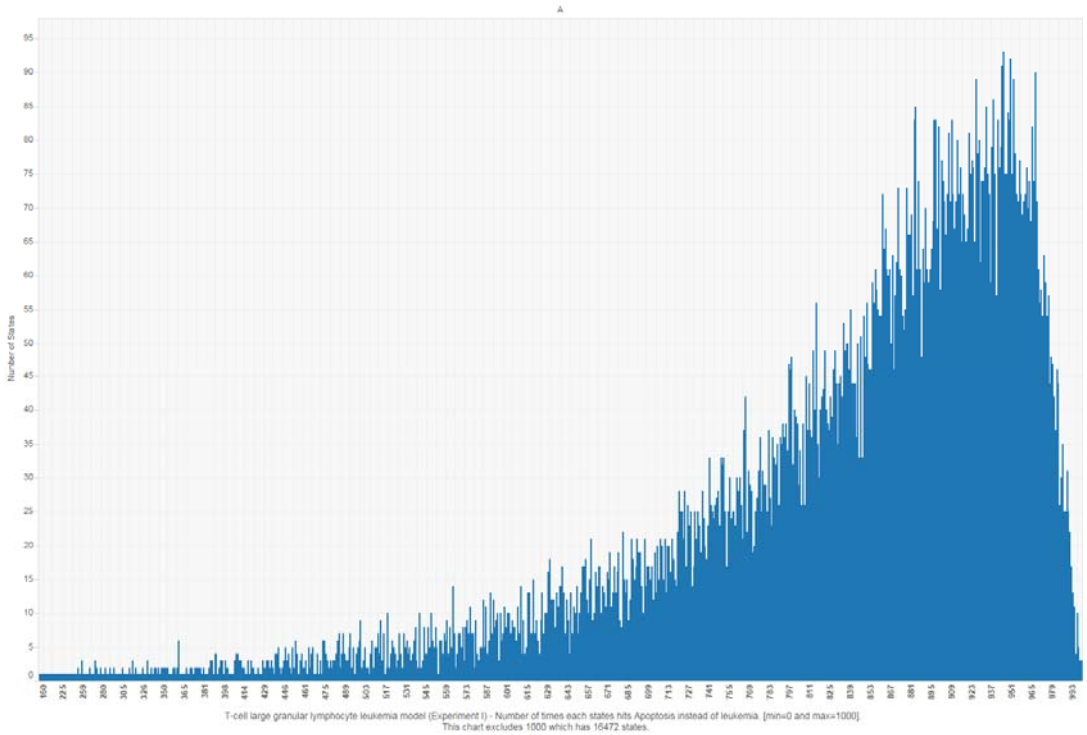

Figure 11: Experiment I – T-LGL with stimuli2 set to false and CD45 set to true.

The machine learning methods used were three decision trees (max splits=4, max split=20, and max split=100), naive Bayesian classifier (using PCA to improve accuracy), and support vector machines (SVM). The machine learning prediction accuracy in experiment I (shown in Table 9) to determine whether a state *s* was predicted to be in $<A, \tau>$ or $\neg<A, \tau>$ was 89.1% for decision trees (max splits=4), 92.4% for decision trees (max splits=20), 94.1% for decision trees (max splits=100), 93.0% for naive Bayesian classifiers, and 96.6% for SVMs. The

chi-square was calculated (Figure 19 show an example of calculation) to determine the p-value. The chi-square calculations were 19221.954 for decision trees (max splits=4), 22334.182 for decision trees (max splits=20), 24416.256 for decision trees (max splits=100), 22736.272 for naive Bayesian classifiers, and 27681.290 for SVMs. In addition, once the chi-square calculation values are known, then the p-values can be derived. For all machine learning methods in all experiments, the p-value<0.0001 was extremely significant for this research. For all other experiments, Table 9 has the values machine learning accuracy, chi-square calculations, and p-values.

Experiment II (Stimuli2=true and CD45=true)

Two attractors were discovered which are shown in Table 1. The healthy attractor *A* (apoptosis) was again used for this study whereas the other cancerous attractor was treated as unhealthy. The state *s* was considered to be in attractor *A* if and only if it's within the threshold range τ, which forms the output classes of $<A, \tau>$. Each state was sampled and the number of hits was recorded to determine its fuzzy membership vector. A histogram chart was created based on these hits in Figure 12. The output class was used to train classifiers using 10 fold cross-validation to make predictions for other states that were not part of the initial subset of states. Table 9 shows the outcome to machine learning accuracy, chi-square calculations, and p-values.

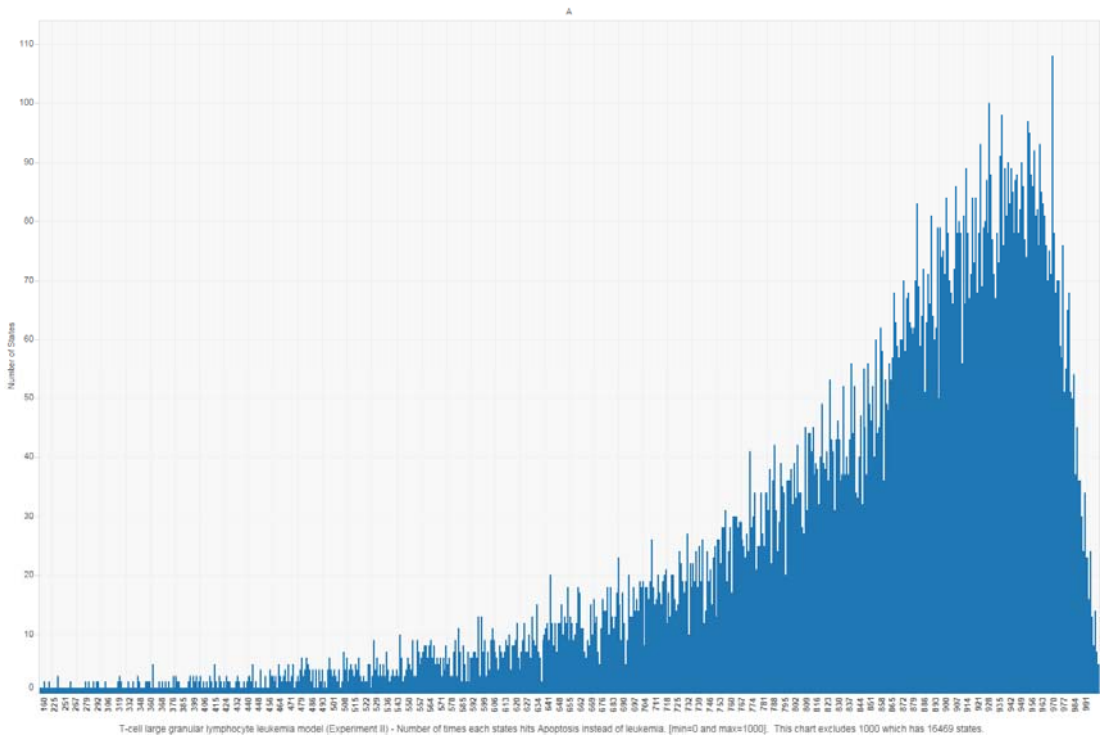

Figure 12: Experiment II – T-LGL with stimuli2 set to true and CD45 set to true.

Experiment III (Stimuli2=false and CD45=false)

Three attractors were discovered which are shown in Table 1. The healthy attractor *A* (apoptosis) was again used whereas the other two cancerous attractors were treated as unhealthy. The state *s* was considered to be in attractor *A* if and only if it's within the threshold range τ, which forms the output classes of $<A, \tau>$. Each state was sampled and the number of hits was recorded to determine its fuzzy membership vector. A histogram chart was created based on these hits in Figure 13. The output class was used to train classifiers using 10 fold cross-validation to make predictions for other states that were not part of the initial subset of states. Table 9 shows the outcome to machine learning accuracy, chi-square calculations, and p-values.

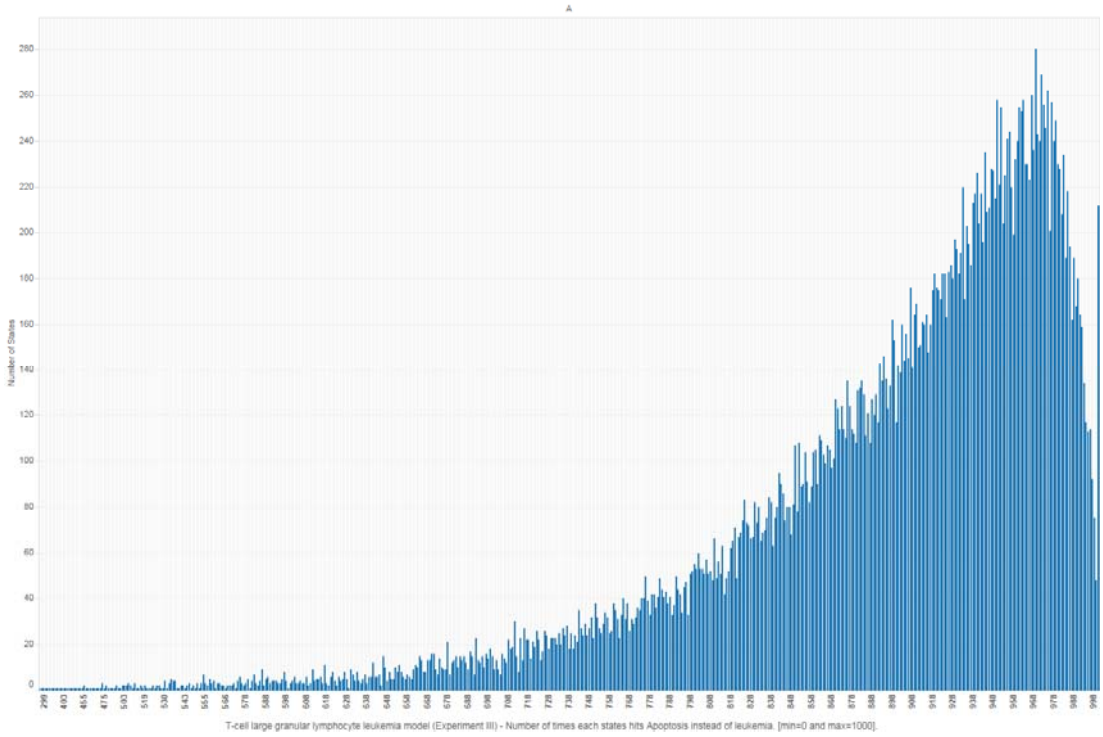

Figure 13: Experiment III – T-LGL with stimuli2 set to false and CD45 set to false.

Experiment IV (Stimuli2=true and CD45=false)

Two attractors were discovered which are also shown in Table 1. The healthy attractor *A* (apoptosis) was again used whereas the other cancerous attractor was treated as unhealthy. A state *s* is considered to be in attractor *A* if and only if it were within the threshold range $\tau = [\tau_{low,}\tau_{high}] = [0.9, 1.0]$, which forms the output classes of $<A, \tau>$ and $\neg<A, \tau>$. Each state was sampled and the number of hits was recorded to determine its fuzzy membership vector. A histogram chart was created based on these hits in Figure 14. The output class was used to train classifiers using 10 fold cross-validation to make predictions for other states that were not part of the initial subset of states. Table 9 shows the outcome to machine learning accuracy, chi-square calculations, and p-values.

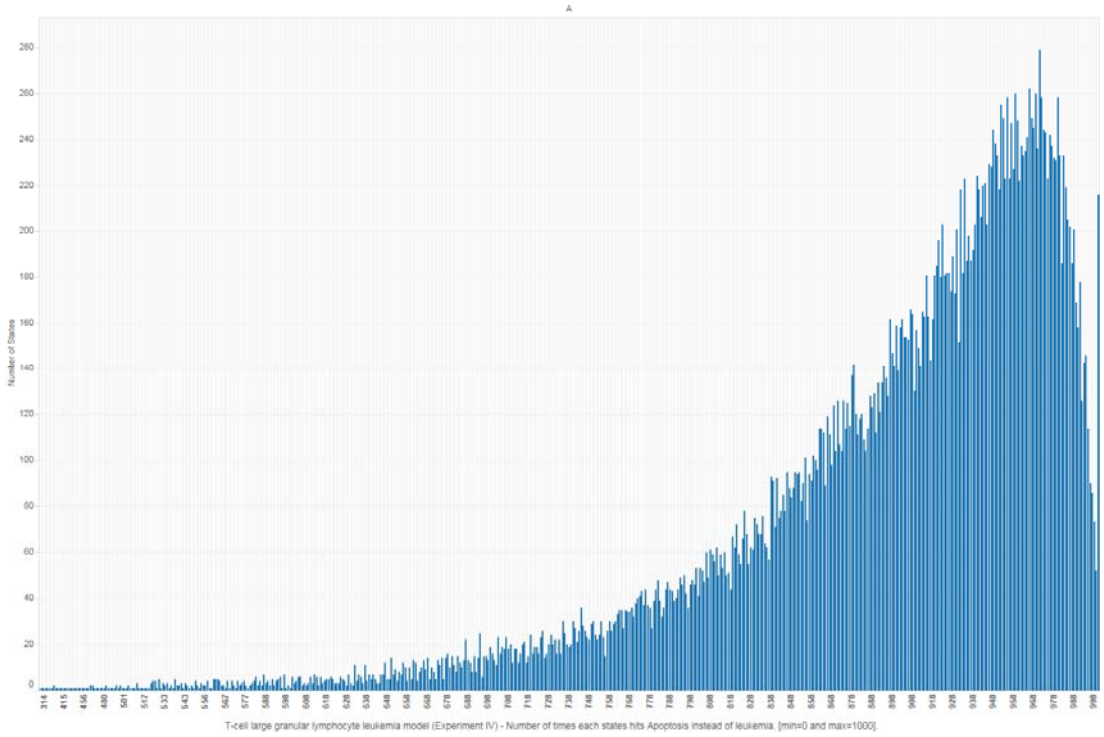

Figure 14: Experiment IV – T-LGL with stimuli2 set to true and CD45 set to false.

Experiments V

Three source input nodes ABH1, ERA1, and AGB1 was set to true (*on*). If these three nodes were allowed to vary in value, then there would be 16 attractors instead of two. During preliminary testing, there were eight attractors for stomatal closure and the other eight for stomatal opening. Thus, forcing the three nodes to remain static throughout state creation allows for the study to focus on two attractors instead of 16. Also, the number of samplings per state was set to N=1000, and the number of transitions allowed is set to r=5000. These parameters, N=1000 and r=5000, also apply to experiments VI - VIII.

Two attractors were discovered which are also shown in Table 4. The basin of attraction for both attractors do not overlap. Each state was sampled *N=1000* times. The transition of each state either hit attractor *A* 0% or 100% of the time. The presence of ABA closes the guard cells and the absent of ABA opens the guard cells. The ABA GRN does not include threshold for ABA building up in guard cells to slowly close nor does it include pathways for guard cells to

absorb ABA to slowly reopen. The healthy attractor *A* (stomatal closure) was used whereas the other attractor *B* (stomatal opening) was treated as unhealthy. Unhealthiness for this experiment, represented by attractor *B*, is turgor pressure and water content are low but the stomatal guard cell remains open.

When turgor pressure is lost, the stoma closes. In angiosperms and gymnosperms, abscisic acid (ABA) is the hormone that triggers the *closing* of the *stomata guard cells* when soil water is insufficient to keep up with transpiration. Turgor pressure is the pressure exerted on a plant cell wall by water passing into the cell by osmosis which forces the plant to stand upright. The reduction of pressure causes the plant to wilt. The guard cells are specialized cells in the epidermis of leaves, stems, and other organs that are used to control gas exchange. ABA binds to receptors at the surface of the plasma membrane of the guard cells.

A state *s* is considered to be in attractor *A* if and only if the probability that they transition to the attractor is within the range $P(s, A) = [\tau_{low,}\tau_{high}] = [0.9, 1.0]$, which forms the output classes of $<A, \tau>$ and $\neg<A, \tau>$. Each state was sampled and the number of hits was recorded to determine its fuzzy membership vector. The output class was used to train classifiers using 10 fold cross-validation to make predictions for other states that were not part of the initial subset of states. Table 9 shows the outcome to machine learning accuracy, chi-square calculations, and p-values.

Experiments VI

For the cardiac development GRN, there are no source input nodes. Six attractors were discovered which are also shown in Table 5. Attractor *A* was used because it had the majority of hits compared to all of the other attractors. A state *s* is considered to be in attractor *A* if and only if it were within the threshold range $\tau = [\tau_{low,}\tau_{high}] = [0.9, 1.0]$, which forms the output classes

of <A, τ> and ¬<A, τ>. Each state was sampled and the number of hits was recorded to determine its fuzzy membership vector. A histogram chart was created based on these hits in Figure 15. The output class was used to train classifiers using 10 fold cross-validation to make predictions for other states that were not part of the initial subset of states. Table 10 shows the outcome to machine learning accuracy, chi-square calculations, and p-values.

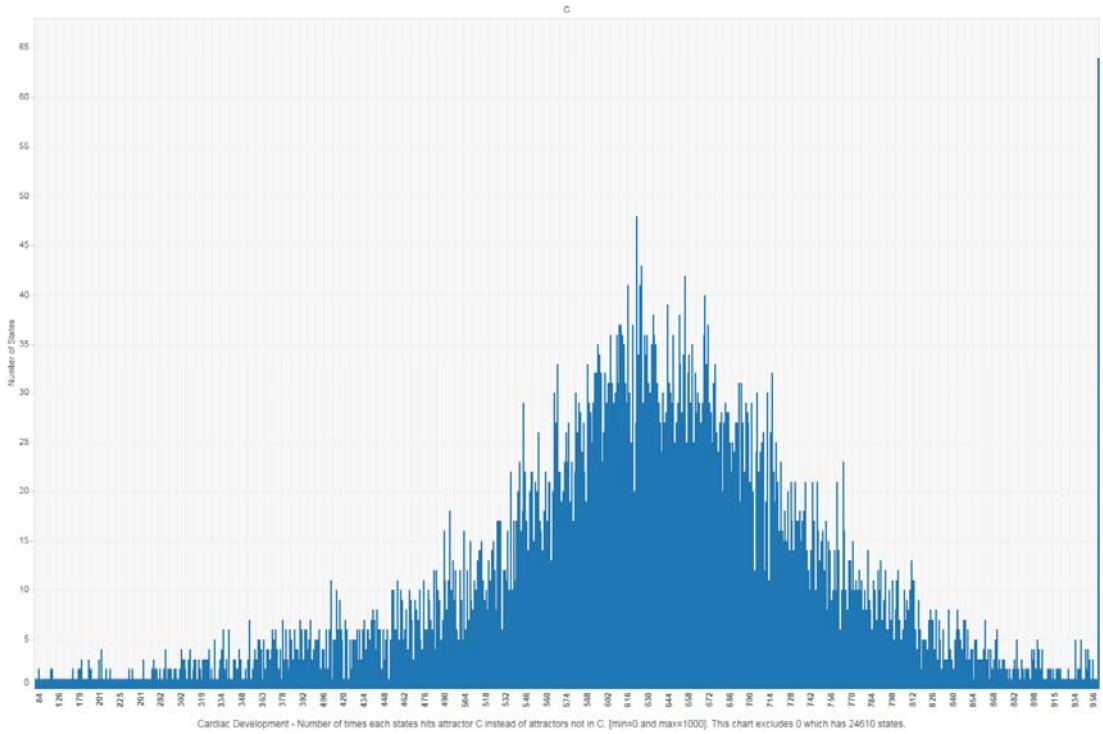

Figure 15: Experiment VI – Cardiac Development.

Experiments VII

For the Mammalian Immune Response to B. Bronchiseptica Infection (Immune Bb) GRN, there are no source input nodes. Ten attractors were discovered which are also shown in Table 6. The healthy attractor A was used because it had the majority of hits compared to all of the other attractors. In addition, the other attractors were treated as unhealthy. A state s is considered to be in attractor A if and only if it were within the threshold range $\tau = [\tau_{low,}\tau_{high}] = [0.9, 1.0]$, which forms the output classes of <A, τ> and ¬<A, τ>. Each state was sampled and the

number of hits was recorded to determine its fuzzy membership vector. A histogram chart was created based on these hits in Figure 16. The output class was used to train classifiers using 10 fold cross-validation to make predictions for other states that were not part of the initial subset of states. Table 10 shows the outcome to machine learning accuracy, chi-square calculations, and p-values.

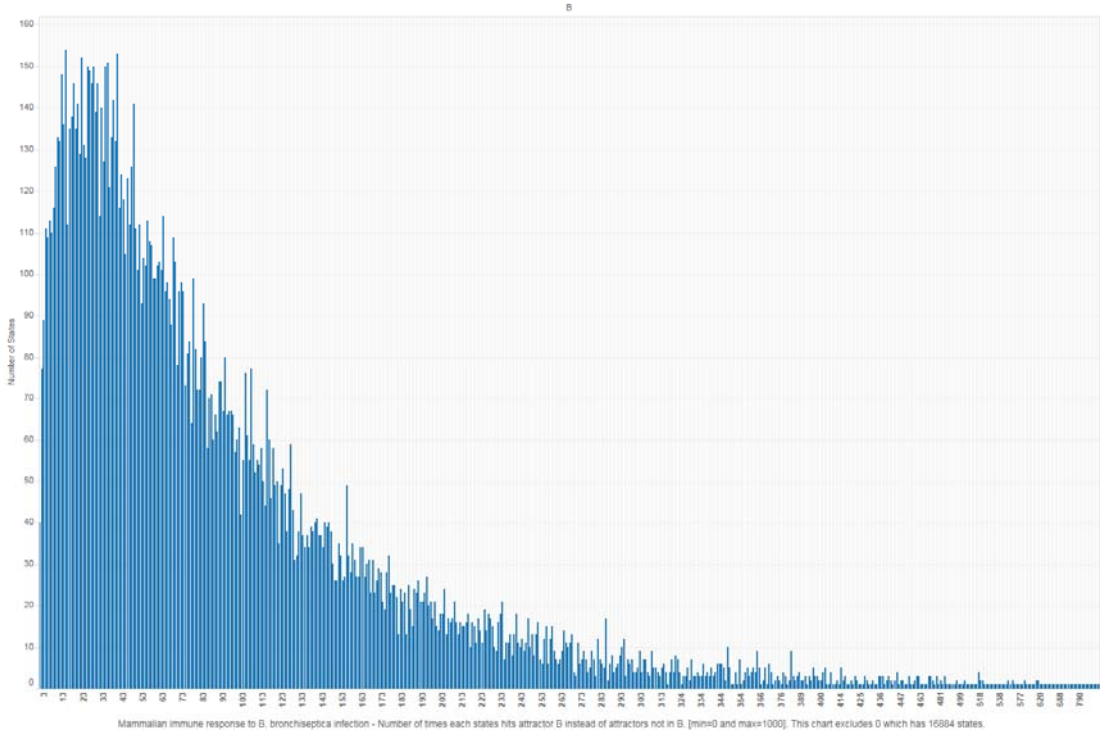

Figure 16: Experiment VII - Mammalian Immune Response to B. Bronchiseptica Infection (Immune Bb).

Experiments VIII

For the mammalian cell cycle GRN, there are no source input nodes. Three attractors were discovered which are also shown in Table 7. The healthy attractor *B* was used because it had the majority of hits compared to all of the other attractors. In addition, the other attractors were treated as unhealthy. A state *s* is considered to be in attractor *B* if and only if it were within the threshold range $\tau = [\tau_{low,}\tau_{high}] = [0.9, 1.0]$, which forms the output classes of <*B*, τ> and ¬<*B*, τ>. Each state was sampled and the number of hits was recorded to determine its fuzzy

membership vector. A histogram chart was created based on these hits in Figure 17. The output class was used to train classifiers using 10 fold cross-validation to make predictions for other states that were not part of the initial subset of states. Table 10 shows the outcome to machine learning accuracy, chi-square calculations, and p-values.

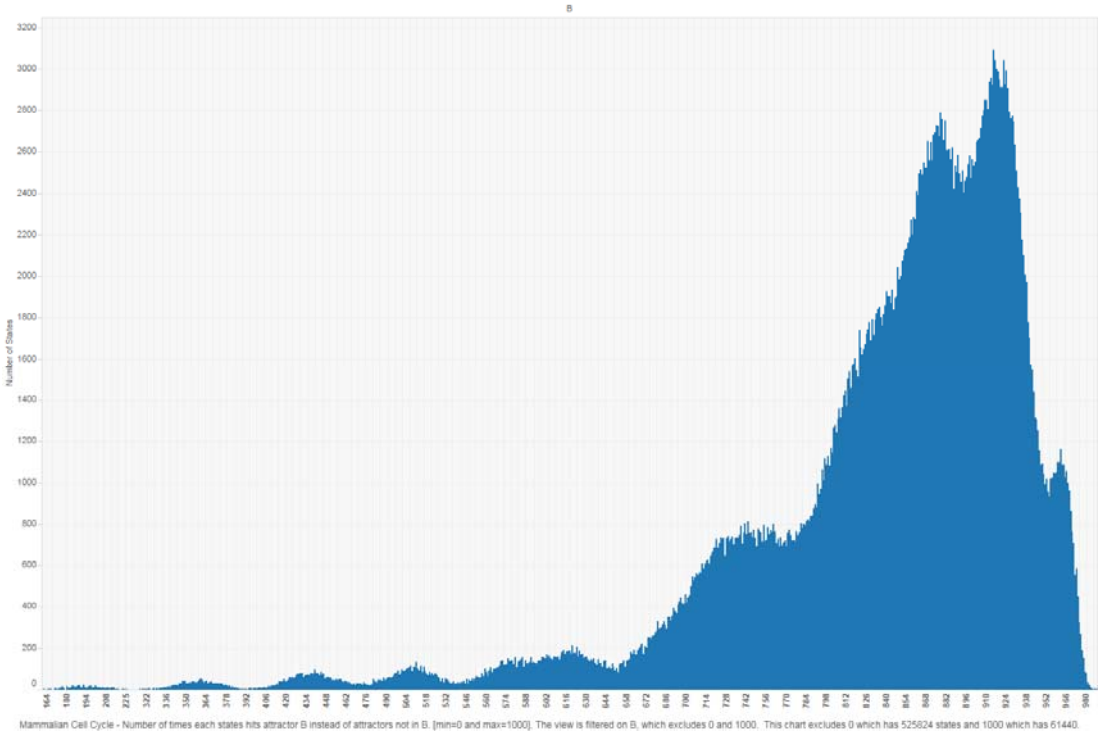

Figure 17: Experiment VIII - Mammalian Cell Cycle

Data Collection and Analysis

This research identified all of the attractors for ABA, Cardiac Development, Immune Bb, T-LGL, and Mammalian Cell Cycle. Tables 4, 5, 6, and 7 comprise the list of attractors for each GRN with the exception of T-LGL. Tables 1 and 2, discussed earlier, provide the list of attractors for T-LGL. The list of attractors was saved to a file for future use to avoid duplicating the search.

Attractors	A (Stomatal Closure)	B (Stomatal Opening)
INPUT NODES		
ABH1	ON	ON
ERA1	ON	ON
AGB1	ON	ON
NODES		
ABA	ON	OFF
GCR1	Oscillates	Oscillates
SphK	ON	OFF
S1P	ON	OFF
GPA1	ON	Oscillates
PLD	ON	Oscillates
PA	ON	Oscillates
pHc	ON	OFF
OST1	ON	OFF
ROP2	ON	Oscillates
Atrboh	ON	OFF
ROS	ON	OFF
H_ATPase	OFF	ON
ABI1	OFF	OFF
RCN1	ON	OFF
NIA12	ON	OFF
NOS	OFF	OFF
NO	OFF	OFF
GC	OFF	OFF
ADPRc	OFF	OFF
cADPR	OFF	OFF
cGMP	OFF	OFF
PLC	OFF	OFF
InsP3	OFF	OFF
InsPK	ON	OFF
InsP6	ON	OFF
CIS	OFF	OFF
Ca2_ATPase	OFF	OFF
Ca2_c	OFF	OFF
AnionEM	ON	OFF
Depolar	ON	Oscillates
CaIM	OFF	OFF
KOUT	ON	Oscillates
KAP	ON	Oscillates
KEV	OFF	OFF
PEPC	OFF	ON
Malate	OFF	ON
RAC1	OFF	ON
Actin	ON	OFF
Closure	ON	OFF

Table 4: The attractors of Abscisic Acid Signaling (ABA). This table shows the state of the nodes for all possible combinations of input signals in the presence and lack of presence of Abscisic acid. The healthy attractor chosen for this study on ABA is the stomatal guard cell closure.

Attractors	A	B	C	D	E	F
INPUT NODES						
NODES						
exogen_BMP2_I	OFF	OFF	OFF	ON	ON	ON
Fgf8	OFF	OFF	OFF	OFF	OFF	ON
Tbx1	OFF	ON	OFF	OFF	OFF	ON
Tbx5	OFF	OFF	ON	OFF	ON	OFF
Foxc1_2	OFF	ON	OFF	OFF	OFF	ON
exogen_canWnt_II	OFF	ON	OFF	OFF	OFF	ON
exogen_BMP2_II	OFF	OFF	OFF	ON	ON	ON
Mesp1	OFF	ON	OFF	OFF	OFF	OFF
Dkk1	OFF	ON	OFF	OFF	OFF	OFF
Bmp2	OFF	OFF	OFF	ON	ON	OFF
Isl1	OFF	ON	OFF	OFF	OFF	ON
canWnt	OFF	ON	OFF	OFF	OFF	ON
GATAs	OFF	ON	ON	OFF	ON	ON
exogen_CanWnt_I	OFF	ON	OFF	OFF	OFF	ON
Nkx2_5	OFF	ON	ON	OFF	ON	ON

Table 5: The attractors of Cardiac Development.

Attractors	A	B	C	D	E	F	G	H	I	J
INPUT NODES										
NODES										
Bb	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
TTSSI	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
TTSSII	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
Oag	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
EC	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
Cab	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON
C	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
AgAb	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
Oab	ON	ON	ON	ON	OFF	OFF	ON	OFF	OFF	OFF
BC	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
PIC	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
IL12I	OFF	OFF	OFF	ON	ON	ON	ON	OFF	ON	OFF
IL12II	OFF	OFF	OFF	ON	ON	ON	ON	OFF	ON	OFF
IL4I	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
IL4II	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
IL10I	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
IL10II	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
IFNgI	OFF	OFF	OFF	ON	ON	ON	ON	OFF	ON	OFF
IFNgII	OFF	OFF	OFF	ON	ON	ON	ON	OFF	ON	OFF
RP	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
DP	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
MPI	OFF	OFF	OFF	ON	ON	ON	ON	OFF	ON	OFF
MPII	OFF	OFF	OFF	ON	ON	ON	ON	OFF	ON	OFF
AP	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
T0	OFF	ON	OFF	ON	ON	ON	ON	OFF	ON	OFF
TrII	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
TrI	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF
Th1II	OFF	OFF	OFF	ON	ON	ON	ON	OFF	ON	OFF
Th1I	OFF	OFF	OFF	ON	ON	ON	ON	OFF	ON	OFF
Th2II	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
h2I	OFF	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
DCI	OFF	ON	OFF	ON	ON	ON	ON	OFF	ON	OFF
DCII	OFF	ON	OFF	ON	ON	ON	ON	OFF	ON	OFF
PH	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Table 6: The attractors of Mammalian Immune Response to B. Bronchiseptica Infection (Immune Bb).

Attractors	A	B	C
INPUT NODES			
NODES			
EGF	OFF	OFF	ON
CDK6	OFF	ON	ON
CDK2	OFF	ON	ON
ErbB1	OFF	OFF	ON
ErbB2_3	OFF	OFF	ON
ERa	OFF	ON	ON
cMYC	OFF	ON	ON
CycE1	OFF	ON	ON
ErbB3	OFF	OFF	ON
CycD1	OFF	ON	ON
p21	OFF	OFF	OFF
IGF1R	OFF	ON	OFF
MEK1	OFF	ON	ON
CDK4	OFF	ON	ON
Akt1	OFF	ON	ON
pRB	OFF	ON	ON
ErbB2	OFF	OFF	ON
ErbB1_2	OFF	OFF	ON
p27	OFF	OFF	OFF
ErbB1_3	OFF	OFF	ON

Table 7: The attractors of Mammalian Cell Cycle.

The list of attractors was collected using Markov Chain Monte Carlo method. The method used a set of randomly chosen start states. Each state was sampled N=1000 times, where each sample is a transition sequence that were run for each sampled state, to determine their fuzzy membership vectors on overlapping attractors. Furthermore, each sampling of a start state involved transitioning to one state then another state until this transition process ends at an attractor. The maximum number of transitions was set to r=5000 to guarantee successful hits to an attractor. The number of hits to each attractor was tallied, where the total number of hits to all attractors for a state equals 1000.

The number of randomly selected states for each GRN was 32768, with the exception of the mammalian cell cycle, which used all 1048576 states. All states also were used for cardiac development, which is a small GRN with only 15 nodes and has a total of 32768 states. Choosing a subset of states reduced the amount of time needed to process the data collection phase of this research.

The data was stored in flat files. Each state was sampled N=1000 times and was associated with the number of hits it had with each attractor, which totaled 1000 hits. Each

attractor was labeled A, B, C, etc., whereas for T-LGL, attractor A was renamed to apoptosis to represent cellular death. Table 5 shows the first few states and their number of hits to each attractor.

States	Apoptosis	Attractor B	Attractor C
s000000000000001100110001010101001001000001001100011010011010	1000	0	0
s000000000000010000011111001100100000110101011100100010010000	1000	0	0
s000000000000011111110001110011111011000111010001000100001100	813	128	59
s000000000000101000011111110001011010000000111001010110011000	842	98	60
s000000000001001001000100111110100110011100110101001100011000	1000	0	0
s000000000001001100000100110100011101001100111111101111111110	1000	0	0
s000000000001010011111001100100011001000101101110110000010000	1000	0	0
s000000000001011011100101000001101000110001010111110001001000	1000	0	0
s000000000001101100011100011101001100111011001111110011000110	1000	0	0
s000000000001101100100010010101100110000111011100110110100110	1000	0	0

Table 8: The first 10 start states from T-LGL1 with *N*=1000 sampling each and the number of times each hit a given attractor.

The structure within a GRN was identified by grouping each state by the number of hits to a specific attractor. A visualization of the structure was created such as those in Figure 11, 12, 13, and 14. The threshold range τ was set for states that reached apoptosis as $\tau = [\tau_{low,}\tau_{high}] = [0.9, 1.0]$ to form the output classes of $<A, \tau>$ and $\neg<A, \tau>$. Classifiers were trained using 10-fold cross validation (and PCA to improve the accuracy of Naive Bayesian classifiers) to predict whether a state reached apoptosis (or not) based on the threshold given.

In initial research, naive Bayesian classifiers did not fare well with data that had high multi-dimensional subspace, such as T-LGL GRN. Principal component analysis (PCA) can solve the problem of high dimensions with dimensionality reduction methods. When PCA was applied to the data before using naive Bayesian classifiers, the prediction accuracy greatly improved, which allowed the classifier to compete closely with SVM and decision trees. However, when PCA was applied before using SVM and decision trees, the classifiers prediction

accuracy was reduced; therefore this study reports only outcomes of naive Bayesian classifiers using PCA and the other two classifiers without using PCA.

Some PCA algorithms, such as the one used in this study, had a pre-processing stage before applying PCA. This preprocessing normalized the mean and variance of the data. The preprocessing algorithm is shown in Figure 18.

The first two steps zeros out the mean of the data.

$$Step\ 1: let\ \mu = \frac{1}{m}\sum_{i=1}^{m} x_{(i)}$$

$$Step\ 2: Replace\ each\ x_{(i)} with\ x_{(i)} - \mu$$

The last two steps rescales each coordinate for normalizing to unit variance.

$$Step\ 3: let\ \sigma_j^2 = \frac{1}{m}\sum_{i} x_{j(i)}$$

$$Step\ 4: Replace\ each\ x_{j(i)}\ with\ \frac{x_{j(i)}}{\sigma_j}$$

Figure 18: Preprocessing the data before PCA to normalize its mean and variance.

Consider a training set, $\boldsymbol{X}$, with the sample mean of each column shifted to zero, where each of the *n* rows represents a different repetition of the experiment, and each of the *p* columns gives a feature for a state *s*, which is a result (0 or 1) from a particular Boolean network function. The transformation is defined by a set of p-dimensional vectors of weights $w_{(k)} = (w_1, \dots, w_p)_{(k)}$ that maps each row vector $x_{(i)} \in \boldsymbol{X}$ to a new vector of principal components $t_{(i)} = (t_1, \dots, t_k)_{(i)}$, given by $t_{k(i)} = x_{(i)} * w_{(k)}$ in such a way that the individual variables of t considered over the data set successively inherit the maximum possible variance from x, with each vector of weights w constrained to be a unit vector.

The first principal component was calculated using the following steps. If $||w|| = 1$, vector $x_{(i)}$ project on w has length $x_{(i)} * w$. To maximize the variance of the projections, we chose a unit-length w so as to maximize $\arg\max_{\|w\|=1}\{\frac{w^T X^T X w}{w^T w}\}$. The first loading vector $w_{(1)}$ has to satisfy $w_{(1)} = \arg\max_{\|w\|=1}\{\sum_i (t_1)^2_{(i)}\} = \arg\max_{\|w\|=1}\{\sum_i (x_{(i)} * w)^2\}$. Rewriting this in matrix form equivalently gives $w_{(1)} = \arg\max_{\|w\|=1}\{\|Xw\|^2\} = \arg\max_{\|w\|=1}\{w^T X^T X w\}$. Since $w_{(1)}$ has been defined to be a unit vector, it equivalently also satisfies $w_{(1)} = \arg\max_{\|w\|=1}\{\frac{w^T X^T X w}{w^T w}\}$. After the first principal component $w_{(1)}$ has been found, then further principal components $w_{(2)}$ to $w_{(k)}$ can be computed.

The *k*th component can be found by subtracting the first $k - 1$ principal components from X, $\hat{X}_k = X - \sum_{s=1}^{k-1} X w_{(s)} w_{(s)}^T$, and then finding the loading vector which extracts the maximum variance from this new data matrix $w_{(k)} = \arg\max_{\|w\|=1}\{\left\|\hat{X}_k w\right\|^2\} = \arg\max_{\|w\|=1}\{\frac{w^T \hat{X}_k^T \hat{X}_k w}{w^T w}\}$. The full principal components decomposition of X can therefore be given as $T = XW$. Finally, for dimensionality reduction, L dimensions can be kept. L in this research is computed by keeping 95% (or more) of the variance within the data.

This research used 10-fold cross-validation in training classifiers to accurately predict outcomes of states. In 10-fold cross-validation, the original sample is randomly partitioned into 10 equal sized subsamples. Of the 10 subsamples, a single subsample is retained as the validation data for testing the model, and the remaining nine subsamples are used as training data. The cross-validation process is then repeated 10 times (the folds), with each of the 10 subsamples used exactly once as the validation data. The 10 results from the folds can then be averaged to produce a single estimation. The advantage of this method over repeated random sub-sampling is

that all observations are used for both training and validation, and each observation is used for validation exactly once.

Findings

T-cell large granular lymphocytic (T-LGL) leukemia, which is a realistic genetic regulatory network (GRN) used in this study, possessed a fuzzy membership structure within its Boolean network state space. Figures 10 - 13 show the fuzzy membership of each start state (randomly chosen) for the apoptosis attractor in the T-LGL GRN. The same randomly chosen start states were used in all four histogram charts. Each state was sampled N=1000 times. During sampling, each time a state reached apoptosis, a bin representing the total was increased by one, forming the histogram charts. Furthermore, each bin represented the number of states with the same totals which determined their fuzzy memberships. Bin count 1000 is not shown in the Figures 10 and 11 because it will flatten all of the other bins in order to fit in the chart. All four histogram charts have Interleukin 15 (IL15), Tax p40 - Human T-lymphotropic virus 1 (TAX), Platelet-derived growth factor beta polypeptide (PDGF), and Antigen stimulation (Stimuli) set to true in the Boolean network. A threshold range was set $\tau = [\tau_{low,}\tau_{high}] = [0.9, 1.0]$, where the bins respectively are between 900 and 1000 hits. In this range, a state was considered to be in the apoptosis attractor and all other states were considered not to be in the apoptosis attractor.

There are four histogram charts (Figures 10-13), one for each experiment (experiment I-IV). In Figure 11, experiment I for T-LGL, new or stronger antigen stimulation (Stimuli2) was set to false and Protein tyrosine phosphatase, receptor type, C (CD45) was set to true within the Boolean network. In Figure 12, experiment II for T-LGL, stimuli2 was set to true and CD45 was also set to true. In Figure 13, experiment III for T-LGL, stimuli2 was set to false and CD45 was also set to false. And, in Figure 14, experiment IV for T-LGL, stimuli2 was set to true and CD45

was also set to false. Machine learning methods will use this range to predict whether other states are within this threshold or not. The importance of knowing the structures and trends that the histograms present helps in the implementation of machine learning methods to accurately predict whether state is either going to reach apoptosis or not.

Boolean networks were used to model GRNs. The T-LGL GRN has many Boolean network nodes and finding a subset of the states' fuzzy membership vectors was time consuming. For instance, each state was sampled *N=1000* times and required maximum *r=5000* transitions per sample to reach an attractor. Furthermore, it was intractable to find fuzzy membership for all states. Machine learning methods were used to predict the fuzzy membership of a state. A subset of states was selected randomly for sampling. Each state was sampled *N*=1000 times to determine its fuzzy membership vector. After the completion of sampling for all states, a threshold was set to determine whether a state can reach apoptosis $\tau = [\tau_{low,}\tau_{high}] = [0.9, 1.0]$. Thus, states within the threshold were classified as positive and are in the apoptosis attractor, whereas states lower than 90% are classified as negative and were not in the apoptosis attractor.

After the classification process, machine learning methods were trained with the sampled data. After training, the machine learning methods were used to predict the outcome for other states that were not part of the initial sampling set. This will allow scientists to choose a set of states to predict their classification without having to go through the sampling process, which can be time consuming. Although, some machine learning methods outperform others in prediction and readability.

A *support vector machine* (SVM) has the best prediction accuracy for predicting the outcome of whether a state will reach apoptosis in T-LGL GRN. Table 9 shows that SVMs led in accuracy for all four experiments in the ML Accuracy column. ML Accuracy is simply the

measure of how many predictions the SVM was able to get correct out of the total number of predictions.

The first column of Table 9 is the GRN being studied and the main attractor, apoptosis, which was used to determine whether a state is in apoptosis or not. Apoptosis was the only attractor chosen because this was the most desired state that helps clinical biologists solves problems with leukemia in white blood cells. The next column in Table 9 is the machine learning methods used for this study. Three types of decision trees were used. The smallest had a maximum of four decision splits to make predictions, the middle decision tree had a maximum of 20 decision splits, and the largest decision tree had a maximum of 100 splits. SVMs and naive Bayesians classifiers use mathematics and statistics for prediction and do not use decision splits. Furthermore, decision trees in this study had output decision rules so that scientist can follow the logic of how the classifier makes its predictions.

The next two columns of Table 9 show the proportion of correct classification for the classifier to predict if a state that does not belong in apoptosis is indeed not in apoptosis or if a state that does belong in apoptosis is indeed in apoptosis. The two columns together are derived from a confusion matrix show in Figure 19. The first, *not in attractor*, is the true negative (TN) divided by total rows and the second, *in attractor*, is the true positive (TP) divided by total rows.

The next three columns of Table 9 are accuracy, precision, and recall. Accuracy is the proportion of correct results that a classifier achieved. For a classifier to accurately predict whether a state reaches apoptosis (or not), precision is the number of correct hits to attractor A divided by the number of all attempts. Recall is the number of correct results divided by the number of results that should have been returned. Chi-squared testing is used to compare two sets of attributes, the predicted values of a classifier and the actual values, to determine whether

the study is statistically significant or not. In Chi-squared testing we compare a set of observed values (O) against a set of expected values (E). The expected values are values that would be expected if there were no association between the predicted and actual values. The calculation of chi^2 is achieved by using the identity $chi^2 = \sum \frac{(O-E)^2}{E}$. If the result is above a given critical threshold value then the study has a statistical significance.

Given a classification rule, it can be determined whether the rule is surprising (i.e. unexpected) or not by determining whether there exists some special relationship between the attributes and the classifier, or that the rule is simply one that is expected assuming a normal chi-squared distribution. A probability value or p-value (p) derived from the critical values of chi-square distribution has shown that all experiments in this study are statistically extremely significant, where $p < 0.0001$. SVM and decision trees both show extreme significances, but SVM has the advantage of performing more accurately whereas decision trees output rules on how its decisions were calculated.

	Predicted 0 (No)	Predicted 1 (Yes)	Row Total
Actual 0 (No)	TN=9540	FP=513	10053
Actual 1 (Yes)	FN=617	TP=22098	22715
Column Total	10157	22611	32768

(a) Confusion Matrix - Observed Values

	Predicted 0 (No)	Predicted 1 (Yes)	Row Total
Actual 0 (No)	(10157)(10053)/(32768) =3116.099	(22611)(10053)/(32768) =6936.901	10053
Actual 1 (Yes)	(10157)(22715)/(32768) =7040.901	(22611)(22715)/(32768) =15674.1	22715
Column Total	10157	22611	32768

(b) Confusion Matrix - Expected Values

Observed (O)	Expected (E)	$(O-E)^2/E$
9540	3116.099	13237.7001
617	7040.901	5860.9692
513	6963.901	5948.8385
22098	15674.100	2632.7822
	Chi-square Total	27681.2900

(c) Chi-square calculation

Critical values of Chi-square Distribution							
DF	p<0.20	p<0.10	p<0.05	p<0.02	p<0.01	p<0.001	p<0.0001
1	chi>1.64	chi>2.71	chi>3.84	chi>5.41	chi>6.63	chi>10.83	chi>15.14

(d) P-value Calculation

Figure 19: The confusion matrix of observed values for the SVM in LGL1-Apoptosis. Steps *a*, *b*, *c*, and *d* show the steps used to arrive at the p-value. It uses true negatives (TN), false negative (FN), false positive (FP), true positive (TP), and degree of freedom (DF) to describe the predicted data.

GRN - Attractor	ML method	Max number of splits	Rules	Not In Attractor (Accuracy)	In Attractor (Accuracy)	Accuracy	Precision	Recall	chi-square	p-value
LGL1 - Apoptosis	Decision Tree	4	Yes	9395/10053 (93.5%)	19768/22715 (87.0%)	89.0%	96.78%	87.03%	19221.954	P < 0.0001
Experiment I	Decision Tree	20	Yes	9168/10053 (91.2%)	21106/22715 (92.9%)	92.4%	95.98%	92.92%	22334.182	P < 0.0001
	Decision Tree	100	Yes	9346/10053 (93.0%)	21480/22715 (94.6%)	94.1%	96.81%	94.56%	24416.256	P < 0.0001
	Naive Bayesian/PCA	N/A	No	8241/10053 (82.0%)	22231/22715 (97.9%)	93.0%	92.46%	97.87%	22736.272	P < 0.0001
	SVM	N/A	No	9540/10053 (94.9%)	22098/22715 (97.3%)	96.6%	97.73%	97.28%	27681.290	P < 0.0001
LGL2 - Apoptosis	Decision Tree	4	Yes	8845/9396 (94.1%)	19905/23372 (85.2%)	87.7%	97.31%	85.17%	17964.867	P < 0.0001
Experiment II	Decision Tree	20	Yes	8306/9396 (88.4%)	21746/23372 (93.0%)	91.7%	95.23%	93.04%	21040.318	P < 0.0001
	Decision Tree	100	Yes	8775/9396 (93.4%)	21923/23372 (93.8%)	93.7%	97.25%	93.80%	23730.494	P < 0.0001
	Naive Bayesian/PCA	N/A	No	7179/9396 (76.4%)	23080/23372 (98.8%)	92.3%	91.24%	98.75%	21501.874	P < 0.0001
	SVM	N/A	No	8899/9396 (94.7%)	22852/23372 (97.8%)	96.9%	97.87%	97.78%	27982.995	P < 0.0001
LGL3 - Apoptosis	Decision Tree	4	Yes	8869/13075 (67.3%)	16220/19693 (82.4%)	76.6%	79.41%	82.36%	8430.694	P < 0.0001
Experiment III	Decision Tree	20	Yes	9242/13075 (70.7%)	17102/19693 (86.8%)	80.4%	81.69%	86.84%	11269.160	P < 0.0001
	Decision Tree	100	Yes	10023/13075 (76.7%)	17405/19693 (88.4%)	83.7%	85.08%	88.38%	14168.910	P < 0.0001
	Naive Bayesian/PCA	N/A	No	10270/13075 (78.5%)	18651/19693 (94.7%)	88.3%	86.93%	94.71%	18651.850	P < 0.0001
	SVM	N/A	No	12086/13075 (92.4%)	18710/19693 (95.0%)	94.0%	94.98%	95.01%	25056.310	P < 0.0001
LGL4 - Apoptosis	Decision Tree	4	Yes	8804/12880 (68.4%)	16392/19888 (82.4%)	76.9%	80.09%	82.42%	8593.638	P < 0.0001
Experiment IV	Decision Tree	20	Yes	9305/12880 (72.2%)	16861/19888 (84.8%)	79.9%	82.51%	84.78%	10827.760	P < 0.0001
	Decision Tree	100	Yes	9864/12880 (76.6%)	17488/19888 (87.9%)	83.5%	85.29%	87.93%	13891.160	P < 0.0001
	Naive Bayesian/PCA	N/A	No	10053/12880 (78.1%)	18891/19888 (95.0%)	88.3%	86.98%	94.99%	18655.140	P < 0.0001
	SVM	N/A	No	11801/12880 (91.6%)	18868/19888 (94.9%)	93.6%	94.59%	94.87%	24550.740	P < 0.0001

Table 9: Machine learning prediction accuracy for the T-LGL GRN.

Scientists that are focused only on accuracy could use SVMs without insight on why such prediction methods are accurate. In all four experiments, SVMs outperformed decision trees and

naive Bayesian classifiers. For SVMs, experiment I was 96.6% accurate, experiment II was 96.9% accurate, experiment III was 93.7% accurate, and experiment IV was 93.6% accurate. Naive Bayesian classifiers performed poorly and needed additional preprocessing to be competitive. Thus, principal component analysis (PCA) was used to boost the performance of a naive Bayesian classifier to get better accuracy in its predictions. PCA was used for SVM and decision trees but their performance in accuracy was reduced. Thus, Table 9 only reflects naive Bayesian classifiers with the usage of PCA. Furthermore, if the analysis of the processes that a machine learning algorithm uses to achieve its predictions is more important than accuracy, decision trees can fulfill that need with rules on how its predictions are calculated.

The decision trees generated rules that are easy to understand. Figure 20 shows a small decision tree's output of rules. This tree had at most four decision splits. The tree of questions are used as a representation language, each node from the tree is either a test about an attribute or a final decision. This four-node decision tree had good accuracy in prediction. For the four-node decision tree (Table 9), experiment I was 89.0% accurate, experiment II was 87.7% accurate, experiment III was 76.6% accurate, and experiment IV was 76.9% accurate. In addition, 20-node and 100-node decision trees showed more accuracy that the four-node counterpart. Theirs were 92.4% and 94.1% respectively for experiment I, 91.7% and 93.7% respectively for experiment II, 80.4% and 83.7% respectively for experiment III, and 79.9% and 83.5% respectively for experiment IV. While accuracy and rules are an important part of this study, output classes can affect the outcome of those accuracies and rules.

```
T-LGL1
node-1  if TBET < 0.5 then node-2 elseif TBET >= 0.5 then node-3 else class = 1
node-2  if Caspase < 0.5 then node-4 elseif Caspase >= 0.5 then node-5 else class = 0
node-3  class = 1
node-4  class = 0
node-5  if DISC < 0.5 then node-6 elseif DISC >= 0.5 then node-7 else class = 1
node-6  class = 0
node-7  class = 1

T-LGL2
node-1  if TBET < 0.5 then node-2 elseif TBET >= 0.5 then node-3 else class = 1
node-2  if Caspase < 0.5 then node-4 elseif Caspase >= 0.5 then node-5 else class = 0
node-3  class = 1
node-4  class = 0
node-5  if DISC < 0.5 then node-6 elseif DISC >= 0.5 then node-7 else class = 1
node-6  class = 0
node-7  class = 1

T-LGL3
node-1  if Caspase < 0.5 then node-2 elseif Caspase >= 0.5 then node-3 else class = 1
node-2  if DISC < 0.5 then node-4 elseif DISC >= 0.5 then node-5 else class = 0
node-3  class = 1
node-4  class = 0
node-5  if TBET < 0.5 then node-6 elseif TBET >= 0.5 then node-7 else class = 1
node-6  class = 0
node-7  class = 1

T-LGL4
node-1  if Caspase < 0.5 then node-2 elseif Caspase >= 0.5 then node-3 else class = 1
node-2  if DISC < 0.5 then node-4 elseif DISC >= 0.5 then node-5 else class = 0
node-3  class = 1
node-4  class = 0
node-5  if Ceramide < 0.5 then node-6 elseif Ceramide >= 0.5 then node-7 else class = 1
node-6  class = 0
node-7  class = 1
```

Figure 20: Decision tree rules with a maximum split of four decision points.

The most useful output classes are finding states that belong to an attractor's true basin of attraction and finding states that are positively within a threshold range τ for attractor A. The output classes are defined as $<A, \tau>$ and $\neg<A, \tau>$. Using output classes allows for grouping and aggregating similar findings within one class from a group of related experiments. The output class consists of several elements. The first element, the attractors from experiment I-IV from the T-LGL GRN is used for this study. As mentioned previously, T-LGL is a large GRN, which was simulated with an asynchronous Boolean network that had 60 nodes. Each node is a transition function that changes the state of the GRN during simulation. The second element, the threshold range τ was set to $\tau = [0.9, 1.0]$. Though, these were the ranges set for this study, it allows for future studies to adjust the ranges accordingly to their scientific needs. Along with the output

classes, there is a subset of states that were within the threshold ranges $<A, \tau>$ and all other states that were outside threshold range $\neg<A, \tau>$. Another complement for the output classes are the rules generated by the decision tree. These rules are easy to understand and were generated from sampling statistics. Together, all of these elements formed the output class which gives enough information without burdening scientist with too much detail.

Summary of Results

Four experiments were executed on the T-LGL GRN using different stimuli in each execution. A subset of states were sampled N=1000 times each. In each sampling process, an individual state would transition into a specified attractor or the state would not transition to that attractor. In addition, a fuzzy membership vector had been formed based on these hits and misses. Furthermore, states with similar fuzzy membership vectors are grouped together to form the bar chart to discover structures in fuzziness. Afterwards, a threshold range was set to further group states with fuzzy membership of $\tau = [\tau_{low,}\tau_{high}] = [0.9, 1.0]$ as the positive group and all other states as the negative group. Machine learning classifiers were trained on these thresholds for future prediction of other states not part of the original selected subset. The accuracy of these classifiers was tested using 10-fold validation and had shown to have extreme statistical significance. This process will be generalized to other GRNs and will be discussed in the next chapter.

Chapter 5

Conclusions, Implications, Recommendations, and Summary

Conclusions

Experiments were conducted with five GRNs: T-LGL, ABA, cardiac, Immune Bb, and mammalian cell cycle. The same process was used to generate start states for training and testing across all GRNs. The number of sampling taken in the T-LGL experiments, N=1000 for each state, was also used for each additional GRN. The attractor that received the most overall hits was selected for each GRN, and the number of hits or misses during sampling to that attractor was collected, and a visual chart based on the number of hits per state was generated. The visualizations of these charts are represented in Figures 10 - 17. Machine learning methods were trained on the collected data to predict whether a state will transition into a healthy attractor or not within threshold range τ. Taking the *chi-square score* and *p-value* will determine the significance of the hypothesis testing for each experiment.

This study uses asynchronous Boolean networks to model and simulate GRNs. An initial subset of states was used to determine fuzzy membership vectors. SVMs, decision trees, and naive Bayesian classifiers was used to discover patterns in the fuzzy membership vectors. The findings from this study suggest these machine learning methods can accurately predict whether a state is within a threshold range τ. Experimenting with additional GRNs has shown in Table 10 that SVMs, decision trees, and naïve Bayesian classifiers can have high accuracy and extreme statistical significance in their predictions. SVMs are the most accurate of the three machine learning classifiers whereas decision trees can produce easy to understand rules such as those listed in appendix A. These rules can be analyzed to determine how decision trees arrive to their conclusions.

GRN - Attractor	ML method	Max number of splits	Rules	Not In Attractor (Accuracy)	In Attractor (Accuracy)	Accuracy	Precision	Recall	chi-square	p-value
LGL1 - Apoptosis	Decision Tree	4	Yes	9395/10053 (93.5%)	19768/22715 (87.0%)	89.0%	96.78%	87.03%	19221.954	P < 0.0001
Experiment I	Decision Tree	20	Yes	9168/10053 (91.2%)	21106/22715 (92.9%)	92.4%	95.98%	92.92%	22334.182	P < 0.0001
	Decision Tree	100	Yes	9346/10053 (93.0%)	21480/22715 (94.6%)	94.1%	96.81%	94.56%	24416.256	P < 0.0001
	Naive Bayesian/PCA	N/A	No	8241/10053 (82.0%)	22231/22715 (97.9%)	93.0%	92.46%	97.87%	22736.272	P < 0.0001
	SVM	N/A	No	9540/10053 (94.9%)	22098/22715 (97.3%)	96.6%	97.73%	97.28%	27681.290	P < 0.0001
LGL2 - Apoptosis	Decision Tree	4	Yes	8845/9396 (94.1%)	19905/23372 (85.2%)	87.7%	97.31%	85.17%	17964.867	P < 0.0001
Experiment II	Decision Tree	20	Yes	8306/9396 (88.4%)	21746/23372 (93.0%)	91.7%	95.23%	93.04%	21040.318	P < 0.0001
	Decision Tree	100	Yes	8775/9396 (93.4%)	21923/23372 (93.8%)	93.7%	97.25%	93.80%	23730.494	P < 0.0001
	Naive Bayesian/PCA	N/A	No	7179/9396 (76.4%)	23080/23372 (98.8%)	92.3%	91.24%	98.75%	21501.874	P < 0.0001
	SVM	N/A	No	8899/9396 (94.7%)	22852/23372 (97.8%)	96.9%	97.87%	97.78%	27982.995	P < 0.0001
LGL3 - Apoptosis	Decision Tree	4	Yes	8869/13075 (67.3%)	16220/19693 (82.4%)	76.6%	79.41%	82.36%	8430.694	P < 0.0001
Experiment III	Decision Tree	20	Yes	9242/13075 (70.7%)	17102/19693 (86.8%)	80.4%	81.69%	86.84%	11269.160	P < 0.0001
	Decision Tree	100	Yes	10023/13075 (76.7%)	17405/19693 (88.4%)	83.7%	85.08%	88.38%	14168.910	P < 0.0001
	Naive Bayesian/PCA	N/A	No	10270/13075 (78.5%)	18651/19693 (94.7%)	88.3%	86.93%	94.71%	18651.850	P < 0.0001
	SVM	N/A	No	12086/13075 (92.4%)	18710/19693 (95.0%)	94.0%	94.98%	95.01%	25056.310	P < 0.0001
LGL4 - Apoptosis	Decision Tree	4	Yes	8804/12880 (68.4%)	16392/19888 (82.4%)	76.9%	80.09%	82.42%	8593.638	P < 0.0001
Experiment IV	Decision Tree	20	Yes	9305/12880 (72.2%)	16861/19888 (84.8%)	79.9%	82.51%	84.78%	10827.760	P < 0.0001
	Decision Tree	100	Yes	9864/12880 (76.6%)	17488/19888 (87.9%)	83.5%	85.29%	87.93%	13891.160	P < 0.0001
	Naive Bayesian/PCA	N/A	No	10053/12880 (78.1%)	18891/19888 (95.0%)	88.3%	86.98%	94.99%	18655.140	P < 0.0001
	SVM	N/A	No	11801/12880 (91.6%)	18868/19888 (94.9%)	93.6%	94.59%	94.87%	24550.740	P < 0.0001
ABA - A	Decision Tree	4	Yes	16332/16332 (100.0%)	16436/16436 (100.0%)	100.0%	100.00%	100.00%	32764.000	P < 0.0001
Experiment V	Decision Tree	20	Yes	16332/16332 (100.0%)	16436/16436 (100.0%)	100.0%	100.00%	100.00%	32764.000	P < 0.0001
	Decision Tree	100	Yes	16332/16332 (100.0%)	16436/16436 (100.0%)	100.0%	100.00%	100.00%	32764.000	P < 0.0001
	Naive Bayesian/PCA	N/A	No	16332/16332 (100.0%)	16436/16436 (100.0%)	100.0%	100.00%	100.00%	32764.000	P < 0.0001
	SVM	N/A	No	16332/16332 (100.0%)	16436/16436 (100.0%)	100.0%	100.00%	100.00%	32764.000	P < 0.0001
Cardiac Development - C	Decision Tree	4	Yes	24576/24576 (100.0%)	8192/8192 (100.0%)	100.0%	100.00%	100.00%	32762.667	P < 0.0001
Experiment VI	Decision Tree	20	Yes	24576/24576 (100.0%)	8192/8192 (100.0%)	100.0%	100.00%	100.00%	32762.667	P < 0.0001
	Decision Tree	100	Yes	24576/24576 (100.0%)	8192/8192 (100.0%)	100.0%	100.00%	100.00%	32762.667	P < 0.0001
	Naive Bayesian/PCA	N/A	No	24576/24576 (100.0%)	8192/8192 (100.0%)	100.0%	100.00%	100.00%	32762.667	P < 0.0001
	SVM	N/A	No	24576/24576 (100.0%)	8192/8192 (100.0%)	100.0%	100.00%	100.00%	32762.667	P < 0.0001
Immune - B	Decision Tree	4	Yes	32114/32114 (100.0%)	1/653 (0.2%)	98.0%	100.00%	0.15%	49.105	P < 0.0001
Experiment VII	Decision Tree	20	Yes	31889/32114 (99.3%)	262/654 (40.1%)	98.1%	53.80%	40.06%	6755.304	P < 0.0001
	Decision Tree	100	Yes	32000/32114 (99.6%)	383/654 (58.6%)	98.8%	77.06%	58.56%	14499.378	P < 0.0001
	Naive Bayesian/PCA	N/A	No	32113/32114 (100.0%)	24/654 (3.7%)	98.1%	96.00%	3.67%	1082.715	P < 0.0001
	SVM	N/A	No	32054/32114 (99.8%)	425/654 (65.0%)	99.1%	87.63%	64.98%	18411.182	P < 0.0001
Mammalian Cell Cycle - B	Decision Tree	4	Yes	524288/524288 (100.0%)	524288/524288 (100.0%)	100.0%	100.00%	100.00%	1048572.000	P < 0.0001
Experiment VIII	Decision Tree	20	Yes	524288/524288 (100.0%)	524288/524288 (100.0%)	100.0%	100.00%	100.00%	1048572.000	P < 0.0001
	Decision Tree	100	Yes	524288/524288 (100.0%)	524288/524288 (100.0%)	100.0%	100.00%	100.00%	1048572.000	P < 0.0001
	Naive Bayesian/PCA	N/A	No	524287/524288 (100.0%)	524284/524288 (100.0%)	100.0%	100.00%	100.00%	1048552.000	P < 0.0001
	SVM	N/A	No	524288/524288 (100.0%)	524288/524288 (100.0%)	100.0%	100.00%	100.00%	1048572.000	P < 0.0001

Table 10: Machine learning prediction accuracy for ABA, cardiac development, Immune Bb, T-LGL, and mammalian cell cycle.

For the GRNs studied in this research, it was seen to hold that realistic GRNs generally possess membership structures within overlapping attractors. For each GRN, the most active attractor was selected to be studied. Figures 10 - 17, mentioned in the previous chapter, shows the structure of the *most active* attractor for each GRN. Each visualization chart verifies the membership structures for a subset of states. The charts reveal that some states within overlapping attractors tend to transition to one attractor more than any other attractor being studied or will only transition to one attractor every time. The attractor that has the most transitions from a group of states is considered the most active attractor for this study. The other

states have less hits for the attractor being studied or simply have no hits in that attractor at all. Thus, some states are within the threshold range τ of being within the attractor where other states are not in that range. For the ABA GRN, a special case has arisen. ABA appears to have two non-overlapping attractors. Thus, a state is either in the healthy attractor *A* (stomatal closure) 100% of the time during sampling or is never in attractor *A*, which means it is in attractor *B* (stomatal opening).

The study also shows that a group of states have similar fuzzy membership to overlapping attractors. For instance, some states invariably transitioned into an attractor *A*, whereas some states never transitioned into the same attractor *A*. Other states share similar percentages in between. The GRNs have states with similar fuzzy membership vectors which can be used to better formulate queries that are solvable by machine learning methods. These cluster of states can allow scientists in future studies to adjust their threshold range τ around states with common fuzziness. These states, grouped together, could be thought of as a super state, and these super states would adopt the fuzzy membership vector that represents the underlying states.

Machine learning can be used to identify accurate fuzzy membership vectors for states within a GRN. In the previous table (Table 10), SVMs ranged from being accurate 93% to 100% for all experiments. All machine learning methods had an extremely high statistical significance where the p-value (p) is less than 0.0001 for all experiments. Thus, using asynchronous Boolean networks as the simulation model can greatly improve the prediction accuracy of machine learning methods. However, this raises the question of whether asynchronous Boolean networks are too constrained as a modeling and simulation process for GRNs.

As previously stated, the most effective machine learning method for accurate prediction accuracy was the SVM, however, the decision tree method produced easy to understand rules.

The SVM that was used makes linear separations of the data. The classifier outperformed all of the other classifiers in accuracy and had extreme statistical significance. Perhaps if a kernel was used to detect non-linearity within the data, the SVM could have improved its accuracy. A drawback on using SVMs is that it's difficult to analyze how the classifier arrived to its prediction. Decision trees aren't as accurate as SVM, but the classifier does produce rules that yield insight into the GRNs structure. These rules for each GRN are listed in appendix A. These rules can be analyzed to see how a classifier arrives to its predictions. The rules can be modified, written in another programming language, and embedded into other application to provide clinical analysis for scientists and useful for gaining insight.

The output classes used for this study are the positive class $<A, \tau>$ and its negation $\neg<A, \tau>$. In this study, the threshold was set to $\tau = [\tau_{low,}\tau_{high}] = [0.9, 1.0]$. The best settings for each GRN could vary to get the most optimal results. But pursuing the best settings for each individual GRN was out of scope for this project. Also, these classes allow clinical biologist to repeat the same experiments in the future, possibly on a different subset of states, for validation and to solve their unique problems.

A result found was that the ABA attractors do not overlap and this lack of overlap is the cause of a completely segregated structure. The fuzzy membership for the subset of states being tested is either 100% or 0% in attractor A. A possible reason for this outcome is that the asynchronous Boolean network model is simulating the opening and closing of guard cells that surrounds stomatal pores is a biological process that acts as a natural survival mechanism in taking in carbon dioxide and/or releasing oxygen. The stomatal guard cells are either opened (or in the process of opening) in the presence of ABA or closed (or in the process of closing) when

there is no ABA present. All fuzziness resides in the opening or closing of the guard cells whereas the attractors that represents opened or closed are absolute concepts.

Another result is that a four-node decision tree has a high degree of accuracy in its prediction rates for all GRNs considered. A possible reason for this unexpected outcome is that specific genes dominate the GRN process which contributes to the high success rate of a four-node decision tree. One example for the four-node decision tree is that T-LGL has four dominant genes. Searching through the rules produced by the four-node decision tree reveals that the four genes with the most influence are TBET, Caspase, DISC, and Ceramide. This outcome suggests that complex GRNs, such as T-LGL, are dominated by a small group of genes for a specific task, in this case, apoptosis. Hence low-depth decision trees are effective.

Four experiments were performed on T-LGL GRN and the study was extended to other GRNs, which was ABA, cardiac, Immune Bb, and mammalian cell cycle. With the other GRNs, the most active attractor was selected for study. After the machine learning classifiers were trained, the study also showed extreme statistical significance on the generalized study of the other GRNs too. However, for the overall methodology of this study, using asynchronous Boolean networks seems to constrain the overall project. The constraints help to structure the methodology to get meaningful results, but there is the possibility that using Boolean networks may not accurately simulate complex systems in cell biology.

Contributions

It is difficult to predict the behavior of GRNs that are of realistic size and complexity. Asynchronous Boolean networks were used to model and simulate GRNs. They were used to efficiently identify a network's set of attractors and predict the likelihood of a state transitioning

into each of these network attractors. Fuzzy membership vectors were used to record the results of these probabilities through the modeling and simulation process.

The goal of this research was to explore methods to discover patterns for a meaningful classification of states in GRNs. The research design took a GRN and a machine learning method as input which produce the output classes $<A, \tau>$ and $\neg<A, \tau>$. SVMs had the highest prediction rates, whereas naive Bayesian classifiers had the highest recall. However, decision trees produced easy to understand rules for prediction procedure. All experiments had an extreme significance with $p_{value} < 0.0001$.

This study has provided a process to model GRNs, which offered insight on how to optimize or control parts of the system by using machine learning techniques against fuzzy membership vectors. Using modeling and simulation of GRNs captured complex interactions between proteins, genes, and biochemicals. Thus, a fast prediction process to determine if a state or set of states has the likelihood of achieving a desired result was produced.

The results had extreme statistical significance with all machine learning methods in all experiments showing a *p-value* of $p < 0.0001$. SVMs had the highest accuracy of all machine learning methods used. Although SVMs was generally higher in precision that all other methods, there was an exception in experiment VII (immune Bb). The decision tree with maximum of four nodes for decision had a *precision=100.0%* and naive Bayesian was next with *precision=96.0%.* Finally, naive Bayesian classifiers had the highest recall in general (ranges between 94.71% and 98.75%), although SVMs weren't that far behind usually with a difference of one percentage point. However, naive Bayesian did poorly on recall with immune Bb with *recall=3.67%.*

This study has shown that the design model has extreme statistical significance when predicting the outcome of a state. Once clinical biologist know the states produced by a

treatment, they can implement this design to predict the outcome of cancer. Although this research produced positive results, some recommendation will be given to improve this study.

Recommendations

While this study has demonstrated the modeling and simulation of a realistic GRN, many opportunities for extending the scope of this study remain. This section presents some of these directions.

Additional Machine Learning Methods

Extreme Gradient Boosting (XGBoost) can enhance the degree of flexibility and scalability as a machine learning method. The XGBoost classifier is implemented with the gradient boosting decision tree algorithm (Chen et al, 2016). Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models in decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. XGBoost utilizes parallelization of tree construction using all of your CPU cores during training, it uses distributed computing for training very large models using a cluster of machines, it manages very large datasets that don't fit into memory, and utilizes cache optimization of data structures and algorithm to make best use of hardware. Implementation of this machine learning method can increase execution speed and model performance during the training phase.

Neural Networks and Deep Learning Methods

In another direction, using an overall model that is not constrained to be interpretable, a more accurate simulation model may be possible. A recurrent neural network model is one possibility. Some advantages of recurrent neural networks are having sigmoid non-linearity that

can be trained through gradient algorithms and comes with a learning algorithm with temporal behavior. These models can take a state *s* as input to produce a state *s′* as output.

Perhaps a more complex recurrent neural network model based on *long and short term memory deep learning* (LSTM) algorithms could more efficiently solve the problem. LSTM blocks or networks are simple recurrent neural network which can be used as a building block of hidden layers for an eventually bigger recurrent neural network. The LSTM block is itself a recurrent network because it contains recurrent connections similar to connections in a conventional recurrent neural network. LSTM algorithms could learn traversal patterns for a start state but with a few exceptions compared to asynchronous Boolean networks. If a certain set of genes are expressed, LSTMs could abstract certain patterns to determine whether a state is likely to lead to an attractor. *Asynchronous* Boolean networks were not constrained to this requirement. Different paths that lead to the current state *s* can have an adverse effect on the probability of reaching the next state. The model, once trained, would have a traversal mechanism built in and a dynamic probability table to determine the changing percentages of reaching the next states.

Developing Better Techniques

A different approach is to develop better methods or techniques that accelerate state transition sequences that allow for larger training sets to be sampled. Larger training set would increase the accuracy of predictions for large GRNs such as T-LGL. Perhaps using parallel process and distributed processes could allow multiple states to be sampled at once. Or, developing better techniques for discovering structures in the GRN's state space to better identify useful output classes.

Additional GRNs

This study used five GRNs, which were ABA, cardiac development, immune, T-LGL, and mammalian cell cycle. This study could be extended to a broader range of GRNs. Perhaps

GRNs based on groups of cells that form coordinated regulation where neighboring cells help regulate themselves as a group is another option. Or using GRNs where organisms mutate with environmental changes.

Summary

The research design takes a GRN and a machine learning method as input and produces output class $<A, \tau>$ and its negation $\neg<A, \tau>$. The classifiers are trained to predict whether a state reaches a healthy attractor or not. The implementation of this methodology had extreme statistical significance where the $p_{value} < 0.0001$ for all experiments, which means that all SVMs, naïve Bayesian classifiers, and decision trees had accurate predictions for all GRNs. Furthermore, the most accurate machine learning method was SVM, while decision trees produced easy to understand rules so clinical biologist can analyze the decision process. The GRN, ABA, does not have overlapping attractors which cause decision trees to produce rules that only had one line. The study also showed that clinical biologist can benefit from this method and can quickly get a response from this approach to solve their problems.

Also discussed is that further research is needed. The implementation could be applied to other GRNs or consider using other machine learning classifiers too. Other suggestions offered are identifying characteristics of GRN that are best suited for a specific classifier. Also, deep learning is another alternative to extend this research.

Appendix A

Decision Tree Rules

ABA

---

simple

---

1 if ABA<0.5 then node 2 elseif ABA>=0.5 then node 3 else 1
2 class = 0
3 class = 1

medium

---

1 if ABA<0.5 then node 2 elseif ABA>=0.5 then node 3 else 1
2 class = 0
3 class = 1

complex

---

1 if ABA<0.5 then node 2 elseif ABA>=0.5 then node 3 else 1
2 class = 0
3 class = 1

Figure 20: ABA (experiment V) decision tree rules.

Cardiac Development
----

simple
----
1 if exogen_BMP2_I<0.5 then node 2 elseif exogen_BMP2_I>=0.5 then node 3 else 0
2 if exogen_CanWnt_I<0.5 then node 4 elseif exogen_CanWnt_I>=0.5 then node 5 else 0
3 class = 0
4 class = 0
5 class = 1

medium
----
1 if exogen_BMP2_I<0.5 then node 2 elseif exogen_BMP2_I>=0.5 then node 3 else 0
2 if exogen_CanWnt_I<0.5 then node 4 elseif exogen_CanWnt_I>=0.5 then node 5 else 0
3 class = 0
4 class = 0
5 class = 1

complex
----
1 if exogen_BMP2_I<0.5 then node 2 elseif exogen_BMP2_I>=0.5 then node 3 else 0
2 if exogen_CanWnt_I<0.5 then node 4 elseif exogen_CanWnt_I>=0.5 then node 5 else 0
3 class = 0
4 class = 0
5 class = 1

Figure 21: Cardiac development (experiment VI) decision tree rules.

Immune Bb

---

simple

```
1  class = 0
```

medium

```
 1  if Bb<0.5 then node 2 elseif Bb>=0.5 then node 3 else 0
 2  if IL4II<0.5 then node 4 elseif IL4II>=0.5 then node 5 else 0
 3  class = 0
 4  class = 0
 5  if IL12II<0.5 then node 6 elseif IL12II>=0.5 then node 7 else 0
 6  if Cab<0.5 then node 8 elseif Cab>=0.5 then node 9 else 0
 7  class = 0
 8  class = 0
 9  if Oab<0.5 then node 10 elseif Oab>=0.5 then node 11 else 0
10  class = 0
11  if DCII<0.5 then node 12 elseif DCII>=0.5 then node 13 else 0
12  class = 0
13  class = 1
```

complex

```
 1  if Bb<0.5 then node 2 elseif Bb>=0.5 then node 3 else 0
 2  if IL4II<0.5 then node 4 elseif IL4II>=0.5 then node 5 else 0
 3  if DCII<0.5 then node 6 elseif DCII>=0.5 then node 7 else 0
 4  if Cab<0.5 then node 8 elseif Cab>=0.5 then node 9 else 0
 5  if IL12II<0.5 then node 10 elseif IL12II>=0.5 then node 11 else 0
 6  class = 0
 7  if IL4II<0.5 then node 12 elseif IL4II>=0.5 then node 13 else 0
 8  class = 0
 9  if Oab<0.5 then node 14 elseif Oab>=0.5 then node 15 else 0
10  if Cab<0.5 then node 16 elseif Cab>=0.5 then node 17 else 0
11  if Cab<0.5 then node 18 elseif Cab>=0.5 then node 19 else 0
12  class = 0
13  if IL12II<0.5 then node 20 elseif IL12II>=0.5 then node 21 else 0
14  class = 0
15  if PIC<0.5 then node 22 elseif PIC>=0.5 then node 23 else 0
16  if DCII<0.5 then node 24 elseif DCII>=0.5 then node 25 else 0
17  if Oab<0.5 then node 26 elseif Oab>=0.5 then node 27 else 0
18  class = 0
19  if Oab<0.5 then node 28 elseif Oab>=0.5 then node 29 else 0
20  if T0<0.5 then node 30 elseif T0>=0.5 then node 31 else 0
```

21 class = 0
22 if IFNgI<0.5 then node 32 elseif IFNgI>=0.5 then node 33 else 0
23 class = 0
24 class = 0
25 if T0<0.5 then node 34 elseif T0>=0.5 then node 35 else 0
26 if T0<0.5 then node 36 elseif T0>=0.5 then node 37 else 0
27 if DCII<0.5 then node 38 elseif DCII>=0.5 then node 39 else 0
28 class = 0
29 if PIC<0.5 then node 40 elseif PIC>=0.5 then node 41 else 0
30 class = 0
31 if DCI<0.5 then node 42 elseif DCI>=0.5 then node 43 else 0
32 if IL10I<0.5 then node 44 elseif IL10I>=0.5 then node 45 else 0
33 class = 0
34 class = 0
35 if DCI<0.5 then node 46 elseif DCI>=0.5 then node 47 else 0
36 class = 0
37 if DCII<0.5 then node 48 elseif DCII>=0.5 then node 49 else 0
38 if IL10I<0.5 then node 50 elseif IL10I>=0.5 then node 51 else 0
39 if T0<0.5 then node 52 elseif T0>=0.5 then node 53 else 1
40 if IL10I<0.5 then node 54 elseif IL10I>=0.5 then node 55 else 0
41 class = 0
42 class = 0
43 if PH<0.5 then node 56 elseif PH>=0.5 then node 57 else 0
44 class = 0
45 if DCI<0.5 then node 58 elseif DCI>=0.5 then node 59 else 0
46 class = 0
47 if Th2II<0.5 then node 60 elseif Th2II>=0.5 then node 61 else 0
48 class = 0
49 if DCI<0.5 then node 62 elseif DCI>=0.5 then node 63 else 0
50 class = 0
51 if IL12I<0.5 then node 64 elseif IL12I>=0.5 then node 65 else 0
52 if IL10I<0.5 then node 66 elseif IL10I>=0.5 then node 67 else 0
53 class = 1
54 class = 0
55 if IL12I<0.5 then node 68 elseif IL12I>=0.5 then node 69 else 0
56 class = 0
57 if AP<0.5 then node 70 elseif AP>=0.5 then node 71 else 0
58 if IL12I<0.5 then node 72 elseif IL12I>=0.5 then node 73 else 0
59 class = 0
60 class = 0
61 if BC<0.5 then node 74 elseif BC>=0.5 then node 75 else 1
62 class = 0
63 if Th2II<0.5 then node 76 elseif Th2II>=0.5 then node 77 else 0
64 if Th1II<0.5 then node 78 elseif Th1II>=0.5 then node 79 else 0
65 class = 0
66 class = 0

67 if TrI<0.5 then node 80 elseif TrI>=0.5 then node 81 else 0
68 if DCII<0.5 then node 82 elseif DCII>=0.5 then node 83 else 0
69 class = 0
70 class = 0
71 if IL10I<0.5 then node 84 elseif IL10I>=0.5 then node 85 else 0
72 if DCII<0.5 then node 86 elseif DCII>=0.5 then node 87 else 1
73 class = 0
74 if Oab<0.5 then node 88 elseif Oab>=0.5 then node 89 else 0
75 class = 1
76 class = 0
77 if BC<0.5 then node 90 elseif BC>=0.5 then node 91 else 1
78 if DP<0.5 then node 92 elseif DP>=0.5 then node 93 else 1
79 class = 0
80 class = 0
81 if IL12I<0.5 then node 94 elseif IL12I>=0.5 then node 95 else 1
82 if TrI<0.5 then node 96 elseif TrI>=0.5 then node 97 else 1
83 class = 0
84 if BC<0.5 then node 98 elseif BC>=0.5 then node 99 else 0
85 class = 0
86 class = 1
87 class = 0
88 class = 0
89 class = 1
90 class = 0
91 class = 1
92 class = 1
93 class = 0
94 class = 1
95 class = 0
96 class = 0
97 class = 1
98 class = 0
99 class = 1

Figure 22: Immune Bb (experiment VII) decision tree rules.

T-LGL – experiment I

---

simple

```
1  if TBET<0.5 then node 2 elseif TBET>=0.5 then node 3 else 1
2  if Caspase<0.5 then node 4 elseif Caspase>=0.5 then node 5 else 0
3  class = 1
4  class = 0
5  if DISC<0.5 then node 6 elseif DISC>=0.5 then node 7 else 1
6  class = 0
7  class = 1
```

medium

```
 1  if TBET<0.5 then node 2 elseif TBET>=0.5 then node 3 else 1
 2  if Caspase<0.5 then node 4 elseif Caspase>=0.5 then node 5 else 0
 3  class = 1
 4  if JAK<0.5 then node 6 elseif JAK>=0.5 then node 7 else 0
 5  if DISC<0.5 then node 8 elseif DISC>=0.5 then node 9 else 1
 6  class = 0
 7  if DISC<0.5 then node 10 elseif DISC>=0.5 then node 11 else 0
 8  if JAK<0.5 then node 12 elseif JAK>=0.5 then node 13 else 0
 9  if JAK<0.5 then node 14 elseif JAK>=0.5 then node 15 else 1
10  class = 0
11  if Fas<0.5 then node 16 elseif Fas>=0.5 then node 17 else 0
12  class = 0
13  if Ceramide<0.5 then node 18 elseif Ceramide>=0.5 then node 19 else 1
14  if Ceramide<0.5 then node 20 elseif Ceramide>=0.5 then node 21 else 1
15  class = 1
16  class = 0
17  if FasT<0.5 then node 22 elseif FasT>=0.5 then node 23 else 1
18  if Fas<0.5 then node 24 elseif Fas>=0.5 then node 25 else 0
19  class = 1
20  if Fas<0.5 then node 26 elseif Fas>=0.5 then node 27 else 1
21  class = 1
22  class = 0
23  class = 1
24  class = 0
25  class = 1
26  class = 0
27  class = 1
```

complex

```
 1  if TBET<0.5 then node 2 elseif TBET>=0.5 then node 3 else 1
 2  if Caspase<0.5 then node 4 elseif Caspase>=0.5 then node 5 else 0
 3  class = 1
 4  if JAK<0.5 then node 6 elseif JAK>=0.5 then node 7 else 0
 5  if DISC<0.5 then node 8 elseif DISC>=0.5 then node 9 else 1
 6  if S1P<0.5 then node 10 elseif S1P>=0.5 then node 11 else 0
 7  if DISC<0.5 then node 12 elseif DISC>=0.5 then node 13 else 0
 8  if JAK<0.5 then node 14 elseif JAK>=0.5 then node 15 else 0
 9  if JAK<0.5 then node 16 elseif JAK>=0.5 then node 17 else 1
10  if Fas<0.5 then node 18 elseif Fas>=0.5 then node 19 else 0
11  class = 0
12  if S1P<0.5 then node 20 elseif S1P>=0.5 then node 21 else 0
13  if Fas<0.5 then node 22 elseif Fas>=0.5 then node 23 else 0
14  if Fas<0.5 then node 24 elseif Fas>=0.5 then node 25 else 0
15  if Ceramide<0.5 then node 26 elseif Ceramide>=0.5 then node 27 else 1
16  if Ceramide<0.5 then node 28 elseif Ceramide>=0.5 then node 29 else 1
17  class = 1
18  class = 0
19  if FasT<0.5 then node 30 elseif FasT>=0.5 then node 31 else 0
20  if Fas<0.5 then node 32 elseif Fas>=0.5 then node 33 else 0
21  class = 0
22  if Ceramide<0.5 then node 34 elseif Ceramide>=0.5 then node 35 else 0
23  if FasT<0.5 then node 36 elseif FasT>=0.5 then node 37 else 1
24  class = 0
25  if S1P<0.5 then node 38 elseif S1P>=0.5 then node 39 else 0
26  if Fas<0.5 then node 40 elseif Fas>=0.5 then node 41 else 0
27  if S1P<0.5 then node 42 elseif S1P>=0.5 then node 43 else 1
28  if Fas<0.5 then node 44 elseif Fas>=0.5 then node 45 else 1
29  if FasT<0.5 then node 46 elseif FasT>=0.5 then node 47 else 1
30  class = 0
31  if NFKB<0.5 then node 48 elseif NFKB>=0.5 then node 49 else 0
32  class = 0
33  if FasT<0.5 then node 50 elseif FasT>=0.5 then node 51 else 0
34  class = 0
35  if FasT<0.5 then node 52 elseif FasT>=0.5 then node 53 else 0
36  if NFKB<0.5 then node 54 elseif NFKB>=0.5 then node 55 else 0
37  if S1P<0.5 then node 56 elseif S1P>=0.5 then node 57 else 1
38  if FasT<0.5 then node 58 elseif FasT>=0.5 then node 59 else 0
39  class = 0
40  if BID<0.5 then node 60 elseif BID>=0.5 then node 61 else 0
41  if SPHK1<0.5 then node 62 elseif SPHK1>=0.5 then node 63 else 1
42  class = 1
43  if FasT<0.5 then node 64 elseif FasT>=0.5 then node 65 else 1
44  if PDGFR<0.5 then node 66 elseif PDGFR>=0.5 then node 67 else 0
```

45 if FasT<0.5 then node 68 elseif FasT>=0.5 then node 69 else 1
46 if NFKB<0.5 then node 70 elseif NFKB>=0.5 then node 71 else 1
47 class = 1
48 class = 0
49 if DISC<0.5 then node 72 elseif DISC>=0.5 then node 73 else 0
50 class = 0
51 if FasL<0.5 then node 74 elseif FasL>=0.5 then node 75 else 0
52 class = 0
53 if S1P<0.5 then node 76 elseif S1P>=0.5 then node 77 else 1
54 class = 0
55 if Ceramide<0.5 then node 78 elseif Ceramide>=0.5 then node 79 else 1
56 class = 1
57 if Ceramide<0.5 then node 80 elseif Ceramide>=0.5 then node 81 else 1
58 class = 0
59 if Ceramide<0.5 then node 82 elseif Ceramide>=0.5 then node 83 else 1
60 class = 0
61 if IAP<0.5 then node 84 elseif IAP>=0.5 then node 85 else 0
62 class = 1
63 if S1P<0.5 then node 86 elseif S1P>=0.5 then node 87 else 0
64 if BID<0.5 then node 88 elseif BID>=0.5 then node 89 else 0
65 class = 1
66 if SPHK1<0.5 then node 90 elseif SPHK1>=0.5 then node 91 else 0
67 class = 0
68 if SPHK1<0.5 then node 92 elseif SPHK1>=0.5 then node 93 else 1
69 class = 1
70 if Fas<0.5 then node 94 elseif Fas>=0.5 then node 95 else 1
71 class = 1
72 class = 0
73 class = 1
74 class = 0
75 class = 1
76 class = 1
77 class = 0
78 class = 0
79 class = 1
80 class = 0
81 class = 1
82 class = 0
83 class = 1
84 class = 1
85 class = 0
86 class = 1
87 class = 0
88 class = 0
89 class = 1
90 class = 1

91 class = 0
92 class = 1
93 class = 0
94 class = 0
95 class = 1

Figure 23: T-LGL (experiment I) decision tree rules.

T-LGL – experiment II

---

simple

---

1 if TBET<0.5 then node 2 elseif TBET>=0.5 then node 3 else 1
2 if Caspase<0.5 then node 4 elseif Caspase>=0.5 then node 5 else 0
3 class = 1
4 class = 0
5 if DISC<0.5 then node 6 elseif DISC>=0.5 then node 7 else 1
6 class = 0
7 class = 1

medium

---

1 if TBET<0.5 then node 2 elseif TBET>=0.5 then node 3 else 1
2 if Caspase<0.5 then node 4 elseif Caspase>=0.5 then node 5 else 0
3 class = 1
4 if JAK<0.5 then node 6 elseif JAK>=0.5 then node 7 else 0
5 if DISC<0.5 then node 8 elseif DISC>=0.5 then node 9 else 1
6 class = 0
7 if DISC<0.5 then node 10 elseif DISC>=0.5 then node 11 else 0
8 if JAK<0.5 then node 12 elseif JAK>=0.5 then node 13 else 0
9 if JAK<0.5 then node 14 elseif JAK>=0.5 then node 15 else 1
10 class = 0
11 if Ceramide<0.5 then node 16 elseif Ceramide>=0.5 then node 17 else 0
12 if S1P<0.5 then node 18 elseif S1P>=0.5 then node 19 else 0
13 if Ceramide<0.5 then node 20 elseif Ceramide>=0.5 then node 21 else 1
14 if Ceramide<0.5 then node 22 elseif Ceramide>=0.5 then node 23 else 1
15 class = 1
16 class = 0
17 class = 1
18 if Fas<0.5 then node 24 elseif Fas>=0.5 then node 25 else 0
19 class = 0

20 if SPHK1<0.5 then node 26 elseif SPHK1>=0.5 then node 27 else 1
21 class = 1
22 if Fas<0.5 then node 28 elseif Fas>=0.5 then node 29 else 1
23 class = 1
24 class = 0
25 class = 1
26 class = 1
27 class = 0
28 class = 0
29 class = 1

complex
----
1 if TBET<0.5 then node 2 elseif TBET>=0.5 then node 3 else 1
2 if Caspase<0.5 then node 4 elseif Caspase>=0.5 then node 5 else 0
3 class = 1
4 if JAK<0.5 then node 6 elseif JAK>=0.5 then node 7 else 0
5 if DISC<0.5 then node 8 elseif DISC>=0.5 then node 9 else 1
6 if S1P<0.5 then node 10 elseif S1P>=0.5 then node 11 else 0
7 if DISC<0.5 then node 12 elseif DISC>=0.5 then node 13 else 0
8 if JAK<0.5 then node 14 elseif JAK>=0.5 then node 15 else 0
9 if JAK<0.5 then node 16 elseif JAK>=0.5 then node 17 else 1
10 if Fas<0.5 then node 18 elseif Fas>=0.5 then node 19 else 0
11 class = 0
12 if S1P<0.5 then node 20 elseif S1P>=0.5 then node 21 else 0
13 if Ceramide<0.5 then node 22 elseif Ceramide>=0.5 then node 23 else 0
14 if S1P<0.5 then node 24 elseif S1P>=0.5 then node 25 else 0
15 if Ceramide<0.5 then node 26 elseif Ceramide>=0.5 then node 27 else 1
16 if Ceramide<0.5 then node 28 elseif Ceramide>=0.5 then node 29 else 1
17 class = 1
18 class = 0
19 if FasT<0.5 then node 30 elseif FasT>=0.5 then node 31 else 0
20 if Fas<0.5 then node 32 elseif Fas>=0.5 then node 33 else 0
21 class = 0
22 if Fas<0.5 then node 34 elseif Fas>=0.5 then node 35 else 0
23 if S1P<0.5 then node 36 elseif S1P>=0.5 then node 37 else 1
24 if Fas<0.5 then node 38 elseif Fas>=0.5 then node 39 else 0
25 class = 0
26 if SPHK1<0.5 then node 40 elseif SPHK1>=0.5 then node 41 else 1
27 if S1P<0.5 then node 42 elseif S1P>=0.5 then node 43 else 1
28 if Fas<0.5 then node 44 elseif Fas>=0.5 then node 45 else 1
29 class = 1
30 class = 0
31 if Ceramide<0.5 then node 46 elseif Ceramide>=0.5 then node 47 else 0
32 class = 0
33 if Ceramide<0.5 then node 48 elseif Ceramide>=0.5 then node 49 else 0

34 class = 0
35 if S1P<0.5 then node 50 elseif S1P>=0.5 then node 51 else 0
36 if FasT<0.5 then node 52 elseif FasT>=0.5 then node 53 else 1
37 if Fas<0.5 then node 54 elseif Fas>=0.5 then node 55 else 0
38 class = 0
39 if FasT<0.5 then node 56 elseif FasT>=0.5 then node 57 else 1
40 if Fas<0.5 then node 58 elseif Fas>=0.5 then node 59 else 1
41 if S1P<0.5 then node 60 elseif S1P>=0.5 then node 61 else 0
42 class = 1
43 if Fas<0.5 then node 62 elseif Fas>=0.5 then node 63 else 1
44 if SPHK1<0.5 then node 64 elseif SPHK1>=0.5 then node 65 else 0
45 if S1P<0.5 then node 66 elseif S1P>=0.5 then node 67 else 1
46 class = 0
47 if DISC<0.5 then node 68 elseif DISC>=0.5 then node 69 else 1
48 class = 0
49 if FasT<0.5 then node 70 elseif FasT>=0.5 then node 71 else 1
50 class = 1
51 class = 0
52 if Fas<0.5 then node 72 elseif Fas>=0.5 then node 73 else 1
53 class = 1
54 class = 0
55 if FasT<0.5 then node 74 elseif FasT>=0.5 then node 75 else 1
56 if BID<0.5 then node 76 elseif BID>=0.5 then node 77 else 0
57 if Ceramide<0.5 then node 78 elseif Ceramide>=0.5 then node 79 else 1
58 if S1P<0.5 then node 80 elseif S1P>=0.5 then node 81 else 1
59 class = 1
60 if Fas<0.5 then node 82 elseif Fas>=0.5 then node 83 else 1
61 class = 0
62 if PDGFR<0.5 then node 84 elseif PDGFR>=0.5 then node 85 else 1
63 class = 1
64 if PDGFR<0.5 then node 86 elseif PDGFR>=0.5 then node 87 else 1
65 class = 0
66 class = 1
67 if SPHK1<0.5 then node 88 elseif SPHK1>=0.5 then node 89 else 1
68 class = 0
69 class = 1
70 class = 0
71 class = 1
72 class = 0
73 class = 1
74 class = 0
75 class = 1
76 class = 0
77 class = 1
78 class = 0
79 class = 1

80 class = 1
81 class = 0
82 class = 0
83 class = 1
84 class = 1
85 class = 0
86 class = 1
87 class = 0
88 class = 1
89 class = 0

Figure 24: T-LGL (experiment II) decision tree rules.

T-LGL – experiment III

---

simple

---

1 if Caspase<0.5 then node 2 elseif Caspase>=0.5 then node 3 else 1
2 if DISC<0.5 then node 4 elseif DISC>=0.5 then node 5 else 0
3 class = 1
4 class = 0
5 if TBET<0.5 then node 6 elseif TBET>=0.5 then node 7 else 1
6 class = 0
7 class = 1

medium

---

1 if Caspase<0.5 then node 2 elseif Caspase>=0.5 then node 3 else 1
2 if DISC<0.5 then node 4 elseif DISC>=0.5 then node 5 else 0
3 if DISC<0.5 then node 6 elseif DISC>=0.5 then node 7 else 1
4 if TBET<0.5 then node 8 elseif TBET>=0.5 then node 9 else 0
5 if TBET<0.5 then node 10 elseif TBET>=0.5 then node 11 else 1
6 if BID<0.5 then node 12 elseif BID>=0.5 then node 13 else 1
7 class = 1
8 class = 0
9 if GZMB<0.5 then node 14 elseif GZMB>=0.5 then node 15 else 0
10 if Ceramide<0.5 then node 16 elseif Ceramide>=0.5 then node 17 else 0
11 class = 1
12 if Ceramide<0.5 then node 18 elseif Ceramide>=0.5 then node 19 else 1
13 class = 1
14 class = 0
15 if MCL1<0.5 then node 20 elseif MCL1>=0.5 then node 21 else 0

16 class = 0
17 if FasT<0.5 then node 22 elseif FasT>=0.5 then node 23 else 1
18 if TBET<0.5 then node 24 elseif TBET>=0.5 then node 25 else 0
19 class = 1
20 class = 1
21 class = 0
22 class = 0
23 class = 1
24 class = 0
25 class = 1

complex
---
1 if Caspase<0.5 then node 2 elseif Caspase>=0.5 then node 3 else 1
2 if DISC<0.5 then node 4 elseif DISC>=0.5 then node 5 else 0
3 if DISC<0.5 then node 6 elseif DISC>=0.5 then node 7 else 1
4 if TBET<0.5 then node 8 elseif TBET>=0.5 then node 9 else 0
5 if TBET<0.5 then node 10 elseif TBET>=0.5 then node 11 else 1
6 if BID<0.5 then node 12 elseif BID>=0.5 then node 13 else 1
7 class = 1
8 if S1P<0.5 then node 14 elseif S1P>=0.5 then node 15 else 0
9 if GZMB<0.5 then node 16 elseif GZMB>=0.5 then node 17 else 0
10 if Ceramide<0.5 then node 18 elseif Ceramide>=0.5 then node 19 else 0
11 if Ceramide<0.5 then node 20 elseif Ceramide>=0.5 then node 21 else 1
12 if Ceramide<0.5 then node 22 elseif Ceramide>=0.5 then node 23 else 1
13 if TBET<0.5 then node 24 elseif TBET>=0.5 then node 25 else 1
14 if FasT<0.5 then node 26 elseif FasT>=0.5 then node 27 else 0
15 class = 0
16 if S1P<0.5 then node 28 elseif S1P>=0.5 then node 29 else 0
17 if MCL1<0.5 then node 30 elseif MCL1>=0.5 then node 31 else 0
18 if Fas<0.5 then node 32 elseif Fas>=0.5 then node 33 else 0
19 if FasT<0.5 then node 34 elseif FasT>=0.5 then node 35 else 1
20 if Fas<0.5 then node 36 elseif Fas>=0.5 then node 37 else 1
21 if FasT<0.5 then node 38 elseif FasT>=0.5 then node 39 else 1
22 if TBET<0.5 then node 40 elseif TBET>=0.5 then node 41 else 0
23 if FasT<0.5 then node 42 elseif FasT>=0.5 then node 43 else 1
24 if IAP<0.5 then node 44 elseif IAP>=0.5 then node 45 else 1
25 class = 1
26 class = 0
27 if Fas<0.5 then node 46 elseif Fas>=0.5 then node 47 else 0
28 if Fas<0.5 then node 48 elseif Fas>=0.5 then node 49 else 0
29 class = 0
30 if BclxL<0.5 then node 50 elseif BclxL>=0.5 then node 51 else 1
31 if BID<0.5 then node 52 elseif BID>=0.5 then node 53 else 0
32 class = 0
33 if FasT<0.5 then node 54 elseif FasT>=0.5 then node 55 else 0

34 if Fas<0.5 then node 56 elseif Fas>=0.5 then node 57 else 0
35 if Fas<0.5 then node 58 elseif Fas>=0.5 then node 59 else 1
36 if GZMB<0.5 then node 60 elseif GZMB>=0.5 then node 61 else 0
37 if RAS<0.5 then node 62 elseif RAS>=0.5 then node 63 else 1
38 if Fas<0.5 then node 64 elseif Fas>=0.5 then node 65 else 1
39 class = 1
40 if Fas<0.5 then node 66 elseif Fas>=0.5 then node 67 else 0
41 if GZMB<0.5 then node 68 elseif GZMB>=0.5 then node 69 else 1
42 if TBET<0.5 then node 70 elseif TBET>=0.5 then node 71 else 1
43 class = 1
44 class = 1
45 if Ceramide<0.5 then node 72 elseif Ceramide>=0.5 then node 73 else 1
46 class = 0
47 if Ceramide<0.5 then node 74 elseif Ceramide>=0.5 then node 75 else 0
48 class = 0
49 if Ceramide<0.5 then node 76 elseif Ceramide>=0.5 then node 77 else 0
50 class = 1
51 if BID<0.5 then node 78 elseif BID>=0.5 then node 79 else 0
52 class = 0
53 if Ceramide<0.5 then node 80 elseif Ceramide>=0.5 then node 81 else 0
54 class = 0
55 if S1P<0.5 then node 82 elseif S1P>=0.5 then node 83 else 1
56 class = 0
57 if S1P<0.5 then node 84 elseif S1P>=0.5 then node 85 else 0
58 if S1P<0.5 then node 86 elseif S1P>=0.5 then node 87 else 1
59 class = 1
60 class = 0
61 if BclxL<0.5 then node 88 elseif BclxL>=0.5 then node 89 else 1
62 class = 1
63 if S1P<0.5 then node 90 elseif S1P>=0.5 then node 91 else 1
64 if GZMB<0.5 then node 92 elseif GZMB>=0.5 then node 93 else 1
65 class = 1
66 class = 0
67 if FasT<0.5 then node 94 elseif FasT>=0.5 then node 95 else 0
68 if Fas<0.5 then node 96 elseif Fas>=0.5 then node 97 else 0
69 class = 1
70 if Fas<0.5 then node 98 elseif Fas>=0.5 then node 99 else 0
71 class = 1
72 if Fas<0.5 then node 100 elseif Fas>=0.5 then node 101 else 0
73 class = 1
74 class = 0
75 class = 1
76 class = 0
77 class = 1
78 class = 0
79 class = 1

80 class = 0
81 class = 1
82 class = 1
83 class = 0
84 class = 1
85 class = 0
86 class = 1
87 class = 0
88 class = 1
89 class = 0
90 class = 1
91 class = 0
92 class = 0
93 class = 1
94 class = 0
95 class = 1
96 class = 0
97 class = 1
98 class = 0
99 class = 1
100 class = 0
101 class = 1

Figure 25: T-LGL (experiment III) decision tree rules.

T-LGL – experiment IV

---

simple

---

1 if Caspase<0.5 then node 2 elseif Caspase>=0.5 then node 3 else 1
2 if DISC<0.5 then node 4 elseif DISC>=0.5 then node 5 else 0
3 class = 1
4 class = 0
5 if Ceramide<0.5 then node 6 elseif Ceramide>=0.5 then node 7 else 1
6 class = 0
7 class = 1

medium

----

1 if Caspase<0.5 then node 2 elseif Caspase>=0.5 then node 3 else 1
2 if DISC<0.5 then node 4 elseif DISC>=0.5 then node 5 else 0
3 if DISC<0.5 then node 6 elseif DISC>=0.5 then node 7 else 1
4 if TBET<0.5 then node 8 elseif TBET>=0.5 then node 9 else 0
5 if Ceramide<0.5 then node 10 elseif Ceramide>=0.5 then node 11 else 1
6 if BID<0.5 then node 12 elseif BID>=0.5 then node 13 else 1
7 class = 1
8 class = 0
9 if GZMB<0.5 then node 14 elseif GZMB>=0.5 then node 15 else 0
10 if TBET<0.5 then node 16 elseif TBET>=0.5 then node 17 else 0
11 if FasT<0.5 then node 18 elseif FasT>=0.5 then node 19 else 1
12 if Ceramide<0.5 then node 20 elseif Ceramide>=0.5 then node 21 else 1
13 class = 1
14 class = 0
15 if MCL1<0.5 then node 22 elseif MCL1>=0.5 then node 23 else 0
16 class = 0
17 class = 1
18 if TBET<0.5 then node 24 elseif TBET>=0.5 then node 25 else 1
19 class = 1
20 class = 0
21 class = 1
22 class = 1
23 class = 0
24 class = 0
25 class = 1

complex

----

1 if Caspase<0.5 then node 2 elseif Caspase>=0.5 then node 3 else 1
2 if DISC<0.5 then node 4 elseif DISC>=0.5 then node 5 else 0
3 if DISC<0.5 then node 6 elseif DISC>=0.5 then node 7 else 1
4 if TBET<0.5 then node 8 elseif TBET>=0.5 then node 9 else 0
5 if Ceramide<0.5 then node 10 elseif Ceramide>=0.5 then node 11 else 1
6 if BID<0.5 then node 12 elseif BID>=0.5 then node 13 else 1
7 class = 1
8 if S1P<0.5 then node 14 elseif S1P>=0.5 then node 15 else 0
9 if GZMB<0.5 then node 16 elseif GZMB>=0.5 then node 17 else 0
10 if TBET<0.5 then node 18 elseif TBET>=0.5 then node 19 else 0
11 if FasT<0.5 then node 20 elseif FasT>=0.5 then node 21 else 1
12 if Ceramide<0.5 then node 22 elseif Ceramide>=0.5 then node 23 else 1
13 if Ceramide<0.5 then node 24 elseif Ceramide>=0.5 then node 25 else 1
14 if FasT<0.5 then node 26 elseif FasT>=0.5 then node 27 else 0
15 class = 0
16 if Fas<0.5 then node 28 elseif Fas>=0.5 then node 29 else 0

17 if MCL1<0.5 then node 30 elseif MCL1>=0.5 then node 31 else 0
18 if Fas<0.5 then node 32 elseif Fas>=0.5 then node 33 else 0
19 if BID<0.5 then node 34 elseif BID>=0.5 then node 35 else 1
20 if TBET<0.5 then node 36 elseif TBET>=0.5 then node 37 else 1
21 if Fas<0.5 then node 38 elseif Fas>=0.5 then node 39 else 1
22 if Fas<0.5 then node 40 elseif Fas>=0.5 then node 41 else 0
23 if FasT<0.5 then node 42 elseif FasT>=0.5 then node 43 else 1
24 if IAP<0.5 then node 44 elseif IAP>=0.5 then node 45 else 1
25 class = 1
26 class = 0
27 if Fas<0.5 then node 46 elseif Fas>=0.5 then node 47 else 0
28 class = 0
29 if S1P<0.5 then node 48 elseif S1P>=0.5 then node 49 else 0
30 if BclxL<0.5 then node 50 elseif BclxL>=0.5 then node 51 else 1
31 if BID<0.5 then node 52 elseif BID>=0.5 then node 53 else 0
32 class = 0
33 if FasT<0.5 then node 54 elseif FasT>=0.5 then node 55 else 0
34 if RAS<0.5 then node 56 elseif RAS>=0.5 then node 57 else 0
35 if PDGFR<0.5 then node 58 elseif PDGFR>=0.5 then node 59 else 1
36 if S1P<0.5 then node 60 elseif S1P>=0.5 then node 61 else 0
37 if Fas<0.5 then node 62 elseif Fas>=0.5 then node 63 else 1
38 if TBET<0.5 then node 64 elseif TBET>=0.5 then node 65 else 1
39 class = 1
40 if TBET<0.5 then node 66 elseif TBET>=0.5 then node 67 else 0
41 if S1P<0.5 then node 68 elseif S1P>=0.5 then node 69 else 1
42 if TBET<0.5 then node 70 elseif TBET>=0.5 then node 71 else 1
43 class = 1
44 class = 1
45 if TBET<0.5 then node 72 elseif TBET>=0.5 then node 73 else 1
46 class = 0
47 if Ceramide<0.5 then node 74 elseif Ceramide>=0.5 then node 75 else 0
48 if Ceramide<0.5 then node 76 elseif Ceramide>=0.5 then node 77 else 0
49 class = 0
50 class = 1
51 if BID<0.5 then node 78 elseif BID>=0.5 then node 79 else 0
52 class = 0
53 if Ceramide<0.5 then node 80 elseif Ceramide>=0.5 then node 81 else 0
54 class = 0
55 if S1P<0.5 then node 82 elseif S1P>=0.5 then node 83 else 1
56 if Fas<0.5 then node 84 elseif Fas>=0.5 then node 85 else 1
57 class = 0
58 class = 1
59 if Fas<0.5 then node 86 elseif Fas>=0.5 then node 87 else 1
60 if Fas<0.5 then node 88 elseif Fas>=0.5 then node 89 else 0
61 class = 0
62 if GZMB<0.5 then node 90 elseif GZMB>=0.5 then node 91 else 1

63 class = 1
64 if S1P<0.5 then node 92 elseif S1P>=0.5 then node 93 else 1
65 class = 1
66 class = 0
67 if BclxL<0.5 then node 94 elseif BclxL>=0.5 then node 95 else 0
68 class = 1
69 class = 0
70 if Fas<0.5 then node 96 elseif Fas>=0.5 then node 97 else 0
71 class = 1
72 if S1P<0.5 then node 98 elseif S1P>=0.5 then node 99 else 0
73 class = 1
74 class = 0
75 class = 1
76 class = 0
77 class = 1
78 class = 0
79 class = 1
80 class = 0
81 class = 1
82 class = 1
83 class = 0
84 class = 0
85 class = 1
86 class = 0
87 class = 1
88 class = 0
89 class = 1
90 class = 0
91 class = 1
92 class = 1
93 class = 0
94 class = 1
95 class = 0
96 class = 0
97 class = 1
98 class = 1
99 class = 0

Figure 26: T-LGL (experiment IV) decision tree rules.

Mammalian Cell Cycle
----

simple
----
1 if EGF<0.5 then node 2 elseif EGF>=0.5 then node 3 else 0
2 class = 0
3 class = 1

medium
----
1 if EGF<0.5 then node 2 elseif EGF>=0.5 then node 3 else 0
2 class = 0
3 class = 1

complex
----
1 if EGF<0.5 then node 2 elseif EGF>=0.5 then node 3 else 0
2 class = 0
3 class = 1

Figure 27: Mammalian cell cycle (experiment VIII) decision tree rules.

References

Albert, I., Thakar, J., Li, S., Zhang, R., & Albert, R. (2008). Boolean network simulations for life scientists. *BioMed Central*.

Asmussen, S. R. (2003). Markov Chains. *Applied Probability and Queues, 51*, 3 - 38. doi: 10.1007/0-387-21525-5_1

Cao, Y. S. G. F. J. (2010, December). A New Approach to Dynamic Fuzzy Modeling of Genetic Regulatory Networks. *NanoBioscience, IEEE Transactions on, 9*(4), 263 - 272. doi: 10.1109/TNB.2010.2082559

Chaves, M., Albert, R., & Sontag, E. (2005). Robustness and fragility of Boolean models for genetic regulatory networks. *J Theor Biol, 235*, 431-449.

Chen, Tianqi, & Guestrin, Carlos (2016). XGBoost: A Scalable Tree Boosting System, arXiv:1603.02754v3 [cs.LG] 10 Jun 2016.

Craddock, T. J. A., Fritsch, P., Mark, A., Rice, J., Rosario, R. M. d., Miller, D. B., Fletcher, M. A., Broderick, G. (2014). A Role for Homeostatic Drive in the Perpetuation of Complex Chronic Illness: Gulf War Illness and Chronic Fatigue Syndrome. *Plos One*. doi: 10.1371/journal.pone.0084839

Deng, H., Runger, G., & Tuv, E. (2011). Bias of Importance Measures for Multi-valued Attributes and Solutions. In T. Honkela, W. Duch, M. Girolami, & S. Kaski (Eds.), *Artificial Neural Networks and Machine Learning – ICANN 2011* (Vol. 6792, pp. 293-300): Springer Berlin Heidelberg.

Domingos, P., & Pazzani, M. (1997). On the Optimality of the Simple Bayesian Classifier Under Zero-One Loss. *Machine Learning, 29*, 103-130.

Esposito, G. (2010). LP-type methods for Optimal Transductive Support Vector Machines. *Dissertation*. Retrieved from http://www.cs.upc.edu/~gesposit/phd3/phd3_esposito.pdf

Faure, A., Naldi, A., Chaouiya, C., & Thieffry, D. (2006). Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics, 22*(14), E124-E131. doi: 10.1093/bioinformatics/btl210

Garg, A., Di Cara, A., Xenarios, I., Mendoza, L., & De Micheli, G. (2008). Synchronous versus asynchronous modeling of gene regulatory networks. *Bioinformatics, 24*(17), 1917-1925. doi: 10.1093/bioinformatics/btn336

Gilks, W. R. (2005). Markov Chain Monte Carlo *Encyclopedia of Biostatistics*: John Wiley & Sons, Ltd.

Gordon, K., & Blobe, G. (2008). Role of transforming growth factor-beta superfaminly signaling pathways in human disease. *Biochim Biophys Acta, 1782*, 197-228.

Grinstead, C. M., & Snell, J. L. (1997). Introduction to Probability. *American Mathematical Society*, Ch. 11: Markov Chains. Retrieved from http://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/Chapter11.pdf

Hallinan, J. S. N. U., Tyne; Bradley, D.R.; Mattick, J.S.; Wiles, J. (2006). Effects of an RNA control layer on the state space of Boolean models of genetic regulatory networks. *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 2551 - 2555. doi: 10.1109/CEC.2006.1688626

Hammersley, J. M., & Handscomb, D. C. (1964). Monte Carlo Methods.

Hong C, Hwang J, Cho K-H, Shin I (2015) An Efficient Steady-State Analysis Method for Large Boolean Networks with High Maximum Node Connectivity. PLoS ONE 10(12): e0145734. https://doi.org/10.1371/journal.pone.0145734

Hopfensitz, M., Müssel, C., Maucher, M., & Kestler, H. (2013). Attractors in Boolean Networks: A Tutorial. *Computational Statistics, 28*(1), 19-36. doi: 10.1007/s00180-012-0324-2

Ikushima, H., & Miyazono, K. (2010). TGFB signalling: a complex web in cancer progression. *Nat Rev Cancer, 10*, 415-424.

Jong, H. D. (2002). Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. *Journal of Computational Biology, 9*, 67-103.

Kauffman, S. A. (1969). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology, 22*(3), 437-467. doi: http://dx.doi.org/10.1016/0022-5193(69)90015-0

Leibiger, I., Brismar, K., & Berggren, P. (2010). Novel aspects on pancreatic beta-cell signal-transduction. *Biochem Biophys Res Commun, 396*, 111-115.

Mavers, M., Ruderman, E., & Perlman, H. (2009). Intracellular signal pathways: potential for therapies. *Curr Rheumatol Rep, 11*, 378-385.

Ng, A. (2015). Support Vector Machines. *Stanford University, CS229 Machine Learning Lecture*(Part V).

Podgorelec, V., Kokol, P., Stiglic, B., & Rozman, I. (2002). Decision Trees: An Overview and Their Use in Medicine. *Journal of Medical Systems, 26*(5), 445-463. doi:10.1023/A:1016409317640

Kohavi, R., and Provost, F. 1998. On Applied Research in Machine Learning. In Editorial for the Special Issue on Applications of Machine Learning and the Knowledge Discovery Process, Columbia University, New York, volume 30.

Saadatpour, A., Wang, R.-S., Liao, A., Liu, X., Loughran, T. P., Albert, I., & Albert, R. (2011). Dynamical and Structural Analysis of a T-Cell Survival Network Identifies Novel Candidate Therapeutic Targets for Large Granular Lymphocyte Leukemia. *PLoS computational biology, 7*(11), e1002267. doi:10.1371/journal.pcbi.1002267

Shah, M., Zhang, R., & Jr, T. L. (2009). Never say die: survival signaling in large granular lymphocyte leukemia. *Clin Lymphoma Myeloma, 9*(3), s244-253.

Shawe-Taylor, J., & Sun, S. (2009). Kernel Methods and Support Vector Machines. *Lecture notes*.

Sokol, L., & Jr., T. L. (2006). Large granular lymphocyte leukemia. *Oncologist, 11*, 263-273.

Sveiczer, A., Novak, B., Mitchison, J. M. (1996). The Size Control of Fission Yeast Revisited. *Journal of Cell Science, 109*, 2947-2957.

Thomas, R. (1991). Regulatory networks seen as asynchronous automata: A logical description. *Journal of Theoretical Biology, 153*(1), 1-23. doi: http://dx.doi.org/10.1016/S0022-5193(05)80350-9

Xiao, Y. (2009). A Tutorial on Analysis and Simulation of Boolean Gene Regulatory Network Models. *Current Genomics, 10*, 511-525.

Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control, 8*(3), 338 - 353.

Zanudo, J., & Albert, R. (2013). An effective network reduction approach to find the dynamical repertoire of discrete dynamic networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science, 23*(2), 025111.

Zhang, R., Shah, M., Yang, J., SB, N., & X, L. (2008). Network model of survival signaling in large granular lymphocyte leukemia. *Proc Natl Acad Sci USA, 105*, 16308-16313.